APPENDIX A:

Test Program, Exercise 1:

Initial Code:

```
; this little program turns an LED light on
mov P1, #00h
mov P1, #01h
loop:
    sjmp loop
```

Code using setb and clr:

```
; this file performs the same task as the test program, but with setb
mov P1, #00h
setb P1.0
loop:
    sjmp loop
```

Most Significant Bit Verification, Exercise 1:

Code:

```
; this file verifies the most significant bit in P1 resides in the 8th LED from the right
mov P1, #00h
mov P1, #80h
loop:
      sjmp loop
```

Light Pattern, Exercise 1:

Code:

```
; this file creates a pattern in the lights that is static
mov P1, #00h
mov P1, #99h
loop:
    sjmp loop
```

Counter Pattern, Exercise 1:

Code:

```
; this file creates a ascending light display
main:
                    mov P1, #0ffh; start off all lit
                    lcall delay

                    mov P1, #00h ; then blank
                    lcall delay

                    setb P1.7
                    lcall delay
                    clr P1.7

                    setb P1.6
                    lcall delay
                    clr P1.6

                    setb P1.5
```

```
lcall delay
clr P1.5

setb P1.4
lcall delay
clr P1.4

setb P1.3
lcall delay
clr P1.3

setb P1.2
lcall delay
clr P1.2

setb P1.1
lcall delay
clr P1.1

setb P1.0
lcall delay
clr P1.0

ljmp main ; restart the process
```

; this function waits for the counter to end before returning

```
delay:
                    mov R0, #0ffh
                    loop:
                      mov R1, #0ffh
                      innerloop:
                            djnz R1, innerloop ; embedded so that delay last for long enough
                            nop
                      djnz R0, loop ; decrements and waits for the counter to end before returning
                      ret
```

Typewriter Program, Exercises 2-3

Initial Code:

```
; the main loop or body of our typewriter program

.org 00h
                    ljmp start ; jumps to the main body of our program (at 100h)

.org 100h
start:
                    lcall init ; starts the serial port
                    loop:
                      lcall getchr  ; gets a character from the PC keyboard
                      lcall sndchr ; echoes the character to the PC screen
                    sjmp loop

init:
; set up serial porr with a 11.0592 Mhz crystal
```

```
; use timer 1 for 9600 baud communications
                    mov tmod, #20h ;set timer 1 to mode 2
                    mov tcon, #40h ; run timer 1
                    mov th1, #-3 ; set 9600 baud
                    mov scon, #50h ; set serial control for 8 bit data
                    ret

getchr:
; this routine receives a character from the PC, transmitted over the serial port
; RI is the same as SCON.0
; the 7 bit ASCII is in the accumulator
                    jnb ri, getchr ; wait till character received
                    mov a, sbuf ; put character in the accumulator
                    anl a, #7fh ; mask off 8th bit, not necessary in ASCII
                    clr ri ; clear 'receive status' flag
                    ret

sndchr:
; this routine transmits a character to the PC using the serial port
; the accumulator holds the character to be sent
; SCON.1 and TI are the same
                    clr scon.1 ; clear the ti complete flag
                    mov sbuf, a ; move a character from the accumulator to the serial buffer
                    txloop:
                      jnb scon.1, txloop ; wait till chr is sent
                    ret
```

Code with crlf and LED bank:

```
; the main loop or body of our typewriter program

.org 00h
                    ljmp start ; jumps to the main body of our program (at 100h)

.org 100h
start:
                    lcall init ; starts the serial port
                    loop:
                      lcall getchr  ; gets a character from the PC keyboard
                      lcall sndchr ; echoes the character to the PC screen
                      djnz R0, loop ; decrements the counter
                      lcall crlf
                    sjmp loop

init:
; set up serial porr with a 11.0592 Mhz crystal
; use timer 1 for 9600 baud communications
                    mov tmod, #20h ;set timer 1 to mode 2
                    mov tcon, #40h ; run timer 1
                    mov th1, #-3 ; set 9600 baud
                    mov scon, #50h ; set serial control for 8 bit data
                    mov R0, #41h ; initializes the chr counter to 65
                    ret

getchr:
; this routine receives a character from the PC, transmitted over the serial port
; RI is the same as SCON.0
```

```
; the 7 bit ASCII is in the accumulator
                    jnb ri, getchr ; wait till character received
                    mov a, sbuf ; put character in the accumulator
                    anl a, #7fh ; mask off 8th bit, not necessary in ASCII
                    clr ri ; clear 'receive status' flag
                    ret

sndchr:
; this routine transmits a character to the PC using the serial port
; the accumulator holds the character to be sent
; SCON.1 and TI are the same
                    clr scon.1 ; clear the ti complete flag
                    mov sbuf, a ; move a character from the accumulator to the serial buffer
                    mov P1, a
                    txloop:
                      jnb scon.1, txloop ; wait till chr is sent
                    ret

crlf:
; this routine handles carriage return and line feed
                    mov a, #0Ah ; makes the linefeed
                    lcall sndchr
                    mov a, #0Dh ; makes the carriage return
                    lcall sndchr
                    mov R0, #41h ; resets the counter to 65
                    ret
```

Calculator Program, Exercises 4-5

Initial Code:

```
; the main loop or body of our calculator program


.org 00h
        ljmp start ; jumps to the main body of our program (at 100h)


.org 100h
start:
        lcall init ; starts the serial port
        loop:
                lcall getNum ; gets the first number
                lcall getNum ; gets the second number
                lcall getOp ; are we +, -, or other
        sjmp loop


getNum:
; this routine handles sotring the three digit number and converting it into hex
        mov R0, #03h ; initializes the chr counter to 3
        innerloop:
                lcall getchr  ; gets a character from the PC keyboard
                lcall sndchr ; echoes the character to the PC screen
                add A, #0d0h ; converts to actual digit
                push acc ; adds the digit to the stack
                djnz R0, innerLoop ; decrements the counter
        lcall crlf ; carriage return and line feed
        ; get the digits from storage and convert to actual value


        pop 00h ; ones place
```

```
        pop 01h ; tens place
        pop 02h ; hundreds place

        mov A, #100d
        mov B, R2
        mul AB ; multiplying the 100s place by 100
        mov R2, A ; R2 holds the result

        mov A, #10d
        mov B, R1
        mul AB ; multiplying the 10s place by 10

        add A, R0 ; add the 1s
        add A, R2 ; add the 100s


        pop 00h ; gets the stored PC from the stack
        pop 01h

        mov R3, A
        push 03h ; add the value to the stack

        push 01h ; restores the PC
        push 00h

        ret

getOp:
```

```
; this routine determines the result and outputs the result in the LED bank
        lcall getchr  ; gets a character from the PC keyboard
        lcall sndchr ; echoes the character to the PC screen
        mov R4, A ; moves the operation into R4

        pop 00h; get the return PC and stores
        pop 01h

        pop 02h ; gets the second number
        pop 03h ; gets the first number
        mov A, R3


        plus:
                cjne R4, #2Bh, minus ; skips to next section if not a plus sign
                add A, R2 ; adds the two numbers
                sjmp endOperations

        minus:
                cjne R4, #2Dh, default ; skips to default if not a minus sign
                clr C ; gets rid of the carry bit so we don't have some issues
                subb A, R2 ; subtracts the second number from the first
                sjmp endOperations

        default:
                mov A, #0h

        endOperations:
```

```
                mov P1, A; shows the result in the LED bank

                push 01h ; restores the return PC
                push 00h

                lcall crlf ; carriage return and line feed
                ret

init:
; set up serial porr with a 11.0592 Mhz crystal
; use timer 1 for 9600 baud communications
        mov tmod, #20h ;set timer 1 to mode 2
        mov tcon, #40h ; run timer 1
        mov th1, #-3 ; set 9600 baud
        mov scon, #50h ; set serial control for 8 bit data
        ret

getchr:
; this routine receives a character from the PC, transmitted over the serial port
; RI is the same as SCON.0
; the 7 bit ASCII is in the accumulator
        jnb ri, getchr ; wait till character received
        mov a, sbuf ; put character in the accumulator
        anl a, #7fh ; mask off 8th bit, not necessary in ASCII
        clr ri ; clear 'receive status' flag
        ret
```

```
sndchr:
; this routine transmits a character to the PC using the serial port
; the accumulator holds the character to be sent
; SCON.1 and TI are the same
        clr scon.1 ; clear the ti complete flag
        mov sbuf, a ; move a character from the accumulator to the serial buffer
        txloop:
                jnb scon.1, txloop ; wait till chr is sent
        ret

crlf:
; this routine handles carriage return and line  feed
        mov a, #0Ah ; makes the linefeed
        lcall sndchr
        mov a, #0Dh ; makes the carriage return
        lcall sndchr
        ret
```

Code with Terminal Display:

```
; the main loop or body of our calculator program with answer in the teraTerm window

.org 00h
        ljmp start ; jumps to the main body of our program (at 100h)

.org 100h
start:
        lcall init ; starts the serial port
        loop:
```

```
            lcall getNum ; gets the first number
            lcall getNum ; gets the second number
            lcall getOp ; are we +, -, or other
        sjmp loop

getNum:
; this routine handles sotring the three digit number and converting it into hex
        mov R0, #03h ; initializes the chr counter to 3
        innerloop:
            lcall getchr  ; gets a character from the PC keyboard
            lcall sndchr ; echoes the character to the PC screen
            add A, #0d0h ; converts to actual digit
            push acc ; adds the digit to the stack
            djnz R0, innerLoop ; decrements the counter
        lcall crlf ; carriage return and line feed
        ; get the digits from storage and convert to actual value

        pop 00h ; ones place
        pop 01h ; tens place
        pop 02h ; hundreds place

        mov A, #100d
        mov B, R2
        mul AB ; multiplying the 100s place by 100
        mov R2, A ; R2 holds the result

        mov A, #10d
        mov B, R1
```

```
        mul AB ; multiplying the 10s place by 10

        add A, R0 ; add the 1s
        add A, R2 ; add the 100s


        pop 00h ; gets the stored PC from the stack
        pop 01h

        mov R3, A
        push 03h ; add the value to the stack

        push 01h ; restores the PC
        push 00h

        ret

getOp:
; this routine determines the result and outputs the result in the LED bank
        lcall getchr  ; gets a character from the PC keyboard
        lcall sndchr ; echoes the character to the PC screen
        mov R4, A ; moves the operation into R4
        lcall crlf ; carriage return and line feed

        pop 00h; get the return PC and stores
        pop 01h

        pop 02h ; gets the second number
```

```
pop 03h ; gets the first number
mov A, R3



plus:
        cjne R4, #2Bh, minus ; skips to next section if not a plus sign
        add A, R2 ; adds the two numbers
        sjmp endOperations

minus:
        cjne R4, #2Dh, default ; skips to default if not a minus sign
        clr C ; gets rid of the carry bit so we don't have some issues
        subb A, R2 ; subtracts the second number from the first
        sjmp endOperations

default:
        mov A, #0h

endOperations:

        mov P1, A; shows the result in the LED bank
        lcall sndresult

        push 01h ; restores the return PC
        push 00h

        lcall crlf ; carriage return and line feed
        ret
```

```
init:
; set up serial porr with a 11.0592 Mhz crystal
; use timer 1 for 9600 baud communications
        mov tmod, #20h ;set timer 1 to mode 2
        mov tcon, #40h ; run timer 1
        mov th1, #-3 ; set 9600 baud
        mov scon, #50h ; set serial control for 8 bit data
        ret

getchr:
; this routine receives a character from the PC, transmitted over the serial port
; RI is the same as SCON.0
; the 7 bit ASCII is in the accumulator
        jnb ri, getchr ; wait till character received
        mov a, sbuf ; put character in the accumulator
        anl a, #7fh ; mask off 8th bit, not necessary in ASCII
        clr ri ; clear 'receive status' flag
        ret

sndchr:
; this routine transmits a character to the PC using the serial port
; the accumulator holds the character to be sent
; SCON.1 and TI are the same
        clr scon.1 ; clear the ti complete flag
        mov sbuf, a ; move a character from the accumulator to the serial buffer
        txloop:
                jnb scon.1, txloop ; wait till chr is sent
```

```
        ret

crlf:
; this routine handles carriage return and line  feed
        mov a, #0Ah ; makes the linefeed
        lcall sndchr
        mov a, #0Dh ; makes the carriage return
        lcall sndchr
        ret

sndresult:
; this routine prints the result to the TeraTerm window
; Acc holds the int value
        mov B, #100d
        div AB ; gets the number of 100s
        lcall displayAscii

        mov A, B
        mov B, #10d
        div AB ; gets the number of 10s
        lcall displayAscii

        mov A, B ; gets the number of 1s
        lcall displayAscii

        lcall crlf  ; carriage return and linefeed

        ret
```

```
displayAscii:
; this subroutine gets the ascii of the digit in A and displays it
        clr C
        subb A, #0d0h
        lcall sndchr
        ret
```

Keypad Interface, Exercise 6

Initial Code:

```
this routine monitors the output from the keypad

.org 00h
        ljmp start ; jumps to the main body of our program (at 100h)

.org 100h
start:
        lcall init ; starts the serial port
        loop:
                jnb P3.2, loop ; checks if a button has been pressed and DA is high

                lcall getNum ; gets the number from the keypad
                lcall sndchr ;displays the output onto the terminal
        sjmp loop

init:
; set up serial port with a 11.0592 Mhz crystal
```

```
; use timer 1 for 9600 baud communications
        mov tmod, #20h ;set timer 1 to mode 2
        mov tcon, #40h ; run timer 1
        mov th1, #-3 ; set 9600 baud
        mov scon, #50h ; set serial control for 8 bit data
        clr P3.3 ; GNDs OE_bar, so that the output is enabled
        ret

getNum:
; this routine gets the value from the keypad and puts in in Acc
        mov A, #4fh ; chosen to range from @ to A-O
        anl A, P1 ; masks off the upper bits
        ret


sndchr:
; this routine transmits a character to the PC using the serial port
; the accumulator holds the character to be sent
; SCON.1 and TI are the same
        clr scon.1 ; clear the ti complete flag
        mov sbuf, a ; move a character from the accumulator to the serial buffer

        txloop:
                jnb scon.1, txloop ; wait till chr is sent
        ret
```

Code with Digit Lookup:

```
; this routine monitors the output from the keypad and changes the keystroke to a digit
```

```
.org 00h
        ljmp start ; jumps to the main body of our program (at 100h)

.org 100h
start:
        lcall init ; starts the serial port
        refresh:
                setb P3.4 ; new buttom press available
        loop:

                jnb P3.2, refresh; checks if a button has been pressed and DA is high
                jnb P3.4, loop ; checks if a new button has been pressed

                lcall getNum ; gets the number from the keypad
                lcall sndchr ;displays the output onto the terminal
        sjmp loop

init:
; set up serial port with a 11.0592 Mhz crystal
; use timer 1 for 9600 baud communications
        mov tmod, #20h ;set timer 1 to mode 2
        mov tcon, #40h ; run timer 1
        mov th1, #-3 ; set 9600 baud
        mov scon, #50h ; set serial control for 8 bit data
        ret

sndchr:
```

```
; this routine transmits a character to the PC using the serial port
; the accumulator holds the character to be sent
; SCON.1 and TI are the same
        clr scon.1 ; clear the ti complete flag
        mov sbuf, a ; move a character from the accumulator to the serial buffer

        txloop:
                jnb scon.1, txloop ; wait till chr is sent
        ret



getNum:
; this routine gets the value from the keypad, converts it to a digit and puts in in Acc
        clr P3.3 ; Gnds output enable
        mov A, #0fh
        anl A, P1 ; masks off the upper bit
        setb P3.3 ; sets output enable high
        lcall getDigit
        clr P3.4; used to determine if a new number has been pressed
        ret

getDigit:
; this routine uses table lookup to get the digit value from the ascii
; accumulator stores 00-0fh value of the keypad press
        inc A
        movc a, @a + pc ; puts value of the transformation into the accumulator
        ret
; table of digits
```

```
        .db 30h, 33h, 32h, 31h ; first row
        .db 30h, 36h, 35h, 34h ; second row
        .db 30h, 39h, 38h, 37h ; third row
        .db 30h, 30h, 30h, 30h ; fourth row
```

Code with Calculator:

```
; our keypad calculator program with answer in the teraTerm window

.org 00h
        ljmp start ; jumps to the main body of our program (at 100h)

.org 100h
start:
        lcall init ; starts the serial port
        loop:
                lcall getNum ; gets the first number
                lcall getNum ; gets the second number
                lcall getOp ; are we +, -, or other
        sjmp loop

digitWait:
        sjmp button
        innerloop:
                setb P3.4 ; new button press available
                button:
                        jnb P3.2, innerloop; checks if a button has been pressed and DA is high
                        jnb P3.4, button; checks if a new button has been pressed
```

```
                ret

getNum:
; this routine handles sotring the three digit number and converting it into hex
        mov R1, #03h ; initializes the chr counter to 3
        numFetch:
                lcall getchr; gets the number from the keypad

                lcall sndchr ; echoes the character to the PC screen

                add A, #0d0h ; transforms to the represented digit

                push acc ; adds the digit to the stack

                djnz R1, numFetch; decrements the counter

        lcall crlf ; carriage return and line feed
        ; get the digits from storage and convert to actual value

        pop 00h ; ones place
        pop 01h ; tens place
        pop 02h ; hundreds place

        mov A, #100d
        mov B, R2
        mul AB ; multiplying the 100s place by 100
        mov R2, A ; R2 holds the result
```

```
        mov A, #10d
        mov B, R1
        mul AB ; multiplying the 10s place by 10

        add A, R0 ; add the 1s
        add A, R2 ; add the 100s

        pop 00h ; gets the stored PC from the stack
        pop 01h

        mov R2, A
        push 02h ; add the value to the stack

        push 01h ; restores the PC
        push 00h

        ret


init:
; set up serial port with a 11.0592 Mhz crystal
; use timer 1 for 9600 baud communications
        mov tmod, #20h ;set timer 1 to mode 2
        mov tcon, #40h ; run timer 1
        mov th1, #-3 ; set 9600 baud
        mov scon, #50h ; set serial control for 8 bit data
        setb P3.4 ; new button press available
        ret
```

```
getchr:
; this routine gets the value from the keypad, converts it to a digit and puts in in Acc
        clr P3.3 ; Gnds output enable
        lcall digitWait
        mov A, #0fh
        anl A, P1 ; masks off the upper bit
        setb P3.3 ; sets output enable high
        lcall getDigit
        clr P3.4; used to determine if a new number has been pressed
        ret

getDigit:
; this routine uses table lookup to get the digit value from the ascii
; accumulator stores 00-0fh value of the keypad press
        inc A
        movc a, @a + pc ; puts value of the transformation into the accumulator
        ret
; table of digits
        .db 2Bh, 33h, 32h, 31h ; first row
        .db 2Dh, 36h, 35h, 34h ; second row
        .db 30h, 39h, 38h, 37h ; third row
        .db 30h, 30h, 30h, 30h ; fourth row

getOp:
; this routine determines the result and outputs the result in the LED bank
        lcall getchr; gets an operation from the keypad
        lcall sndchr ; echoes the character to the PC screen
```

```
mov R4, A ; moves the operation into R4
lcall crlf ; carriage return and line feed

pop 00h; get the return PC and stores
pop 01

pop 02h ; gets the second number
pop 03h ; gets the first number
mov A, R3 ; moves the first number into the accumulator

plus:
        cjne R4, #2Bh, minus ; skips to next section if not a plus sign
        add A, R2 ; adds the two numbers
        sjmp endOperations

minus:
        cjne R4, #2Dh, default ; skips to default if not a minus sign
        clr C ; gets rid of the carry bit so we don't have some issues
        subb A, R2 ; subtracts the second number from the first
        sjmp endOperations

default:
        mov A, #0h

endOperations:

        lcall sndresult
```

```
        push 01h ; restores the return PC
        push 00h

        lcall crlf ; carriage return and line feed
        ret

sndresult:
; this routine prints the result to the TeraTerm window
; Acc holds the int value
        mov B, #100d
        div AB ; gets the number of 100s
        lcall displayAscii

        mov A, B
        mov B, #10d
        div AB ; gets the number of 10s
        lcall displayAscii

        mov A, B ; gets the number of 1s
        lcall displayAscii

        lcall crlf  ; carriage return and linefeed

        ret

displayAscii:
; this subroutine gets the ascii of the digit in A and displays it
        clr C
```

```
        subb A, #0d0h
        lcall sndchr
        ret

sndchr:
; this routine transmits a character to the PC using the serial port
; the accumulator holds the character to be sent
; SCON.1 and TI are the same
        clr scon.1 ; clear the ti complete flag
        mov sbuf, a ; move a character from the accumulator to the serial buffer
        txloop:
                jnb scon.1, txloop ; wait till chr is sent
        ret

crlf:
; this routine handles carriage return and line  feed
        mov a, #0Ah ; makes the linefeed
        lcall sndchr
        mov a, #0Dh ; makes the carriage return
        lcall sndchr
        ret
```