.cdecls C,LIST,"msp430fg4618.h"

**.sect** ".const"

.bss label, 4

**.word** 0x1234

strg2 .string "Laboratory #2 for EEL4742 embedded Systems"

**.byte** 0x0d,0x0a ;

**.byte** 0x00

**.text**

**.global** \_START

;----------------------------------------------------------------

**START** **mov.w** #300h,SP

StopWDT **mov.w** #WDTPW+WDTHOLD,&WDTCTL

**call** #Init\_UART

Mainloop

;call #part1

**call** #part2

;call #part3

EXIT **jmp** Mainloop

part1

**call** #INCHAR\_UART

**mov.b** R4, R7 ; R7 will represent first

**call** #INCHAR\_UART

; call #OUTA\_UART

**mov.b** R4, R8 ; R7 will represent second

; call #OUTA\_UART

convertFirst **cmp.b** #0x30, R7

**jge** next11

next11 **cmp.b** #0x39, R7

**jl** addop1

**cmp.b** #0x39, R7

**jeq** addop1

**jmp** next21

addop1 **sub.b** #0x30, R7

**jmp** cont

next21 **cmp.b** #0x41, R7

**jge** next31

next31 **cmp.b** #0x46, R7

**jl** subtract1

next51 **cmp.b** #0x46, R7

**jeq** subtract1

subtract1 **sub.b** #0x37, R7

**jmp** cont

cont

convertSec **cmp.b** #0x30, R8

**jge** next112

next112 **cmp.b** #0x39, R8

**jl** addop12

**cmp.b** #0x39, R8

**jeq** addop12

**jmp** next212

addop12 **sub.b** #0x30, R8

**jmp** cont2

next212 **cmp.b** #0x41, R8

**jge** next312

next312 **cmp.b** #0x46, R8

**jl** subtract12

next512 **cmp.b** #0x46, R8

**jeq** subtract12

subtract12 **sub.b** #0x37, R8

**jmp** cont2

cont2

**rla** R7

**rla** R7

**rla** R7

**rla** R7

**add.b** R7, R8

**mov.b** R8, R4

**call** #OUTA\_UART

Exit1 **ret**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; input S output 53

part2

;char second = c & 0x0F; //0x15 = 1111

; char first = c;

;first = c & 0xF0;

;first = first >> 4;

;OUTA\_UART(getHexVal(first));

;OUTA\_UART(getHexVal(second));

**call** #INCHAR\_UART

**mov.b** R4, R7 ; R7 will represent second, R8 will represent first

**mov.b** R7, R8

**and.b** #0x0F,R7

**and.b** #0xf0, R8

**rra** R8

**rra** R8

**rra** R8

**rra** R8

;;; good until this

convertFirst2 **cmp.b** #0x30, R7

**jge** next1123

next1123 **cmp.b** #0x39, R7

**jl** addop123

**cmp.b** #0x39, R7

**jeq** addop123

**jmp** next2123

addop123 **add.b** #0x37, R7

**jmp** cont3

next2123 **cmp.b** #0x41, R7

**jge** next3123

next3123 **cmp.b** #0x46, R7

**jl** subtract123

next5123 **cmp.b** #0x46, R7

**jeq** subtract123

subtract123 **add.b** #0x37, R7

**jmp** cont3

cont3

convertSec3 **cmp.b** #0x30, R8

**jge** next11234

next11234 **cmp.b** #0x39, R8

**jl** addop1234

**cmp.b** #0x39, R8

**jeq** addop1234

**jmp** next21234

addop1234 **add.b** #0x30, R8

**jmp** cont234

next21234 **cmp.b** #0x41, R8

**jge** next31234

next31234 **cmp.b** #0x46, R8

**jl** subtract1234

next51234 **cmp.b** #0x46, R8

**jeq** subtract1234

subtract1234 **add.b** #0x37, R8

**jmp** cont234

cont234

**mov.b** R8, R4

**call** #OUTA\_UART

**mov.b** R7, R4

**call** #OUTA\_UART

part2exit **ret**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

part3

**call** #INCHAR\_UART

**call** #OUTA\_UART

CHECKUPPER **cmp.b** #0x41, R4

**jge** next1

next1 **cmp.b** #0x5A, R4

**jl** addop

**cmp.b** #0x5A, R4

**jeq** addop

**jmp** next2

addop **add.b** #0x20, R4

**call** #OUTA\_UART

**jmp** Exit

next2 **cmp.b** #0x61, R4

**jge** next3

next3 **cmp.b** #0x7A, R4

**jl** subtract

next5 **cmp.b** #0x7A, R4

**jeq** subtract

subtract **sub.b** #0x20, R4

**call** #OUTA\_UART

**jmp** Exit

Exit **ret**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

OUTA\_UART

;----------------------------------------------------------------

; prints to the screen the ASCII value stored in register 4 and

; uses register 5 as a temp value

;----------------------------------------------------------------

; IFG2 register (1) = 1 transmit buffer is empty,

; UCA0TXBUF 8 bit transmit buffer

; wait for the transmit buffer to be empty before sending the

; data out

**push** R5

lpa **mov.b** &IFG2,R5

**and.b** #0x02,R5

**cmp.b** #0x00,R5

**jz** lpa

; send the data to the transmit buffer UCA0TXBUF = A;

**mov.b** R4,&UCA0TXBUF

**pop** R5

**ret**

INCHAR\_UART

;----------------------------------------------------------------

; returns the ASCII value in register 4

;----------------------------------------------------------------

; IFG2 register (0) = 1 receive buffer is full,

; UCA0RXBUF 8 bit receive buffer

; wait for the receive buffer is full before getting the data

**push** R5

lpb **mov.b** &IFG2,R5

**and.b** #0x01,R5

**cmp.b** #0x00,R5

**jz** lpb

**mov.b** &UCA0RXBUF,R4

**pop** R5

; go get the char from the receive buffer

**ret**

Init\_UART

;----------------------------------------------------------------

; Initialization code to set up the uart on the experimenter board to 8 data,

; 1 stop, no parity, and 9600 baud, polling operation

;----------------------------------------------------------------

;P2SEL=0x30;

; transmit and receive to port 2 b its 4 and 5

**mov.b** #0x30,&P2SEL

; Bits p2.4 transmit and p2.5 receive UCA0CTL0=0

; 8 data, no parity 1 stop, uart, async

**mov.b** #0x00,&UCA0CTL0

; (7)=1 (parity), (6)=1 Even, (5)= 0 lsb first,

; (4)= 0 8 data / 1 7 data, (3) 0 1 stop 1 / 2 stop, (2-1) --

; UART mode, (0) 0 = async

; UCA0CTL1= 0x41;

**mov.b** #0x41,&UCA0CTL1

; select ALK 32768 and put in software reset the UART

; (7-6) 00 UCLK, 01 ACLK (32768 hz), 10 SMCLK, 11 SMCLK

; (0) = 1 reset

;UCA0BR1=0;

; upper byte of divider clock word

**mov.b** #0x00,&UCA0BR1

;UCA0BR0=3; ;

; clock divide from a clock to bit clock 32768/9600 = 3.413

**mov.b** #0x03,&UCA0BR0

; UCA0BR1:UCA0BR0 two 8 bit reg to from 16 bit clock divider

; for the baud rate

;UCA0MCTL=0x06;

; low frequency mode module 3 modulation pater used for the bit

; clock

**mov.b** #0x06,&UCA0MCTL

;UCA0STAT=0;

; do not loop the transmitter back to the receiver for echoing

**mov.b** #0x00,&UCA0STAT

; (7) = 1 echo back trans to rec

; (6) = 1 framing, (5) = 1 overrun, (4) =1 Parity, (3) = 1 break

; (0) = 2 transmitting or receiving data

;UCA0CTL1=0x40;

; take UART out of reset

**mov.b** #0x40,&UCA0CTL1

;IE2=0;

; turn transmit interrupts off

**mov.b** #0x00,&IE2

; (0) = 1 receiver buffer Interrupts enabled

; (1) = 1 transmit buffer Interrupts enabled

;----------------------------------------------------------------

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;----------------------------------------------------------------

; IFG2 register (0) = 1 receiver buffer is full, UCA0RXIFG

; IFG2 register (1) = 1 transmit buffer is empty, UCA0RXIFG

; UCA0RXBUF 8 bit receiver buffer, UCA0TXBUF 8 bit transmit

; buffer

**ret**

;----------------------------------------------------------------

; Interrupt Vectors

;----------------------------------------------------------------

**.sect** ".reset" ; MSP430 RESET Vector

**.short** **START** ;

.end