; final assembly code for lab #5 kelsey cameron

.cdecls C,LIST,"msp430fg4618.h"

.sect ".const"

.bss label, 4

.word 0x1234

.text

.global \_START

.bss myarray, 32

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

START mov.w #300h,SP

StopWDT mov.w #WDTPW+WDTHOLD,&WDTCTL

; uncomment the lab stuff above this line

; delete below this line for lab

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

; MSP430 Assembler Code Template for use with TI Code Composer Studio

;

;

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

; .cdecls C,LIST,"msp430f5529.h" ; Include device header file

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

; .def RESET ; Export program entry-point to

; make it known to linker.

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

; .text ; Assemble into program memory.

; .retain ; Override ELF conditional linking

; and retain current section.

; .retainrefs ; And retain any sections that have

; references to current section.

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

;RESET mov.w #\_\_STACK\_END,SP ; Initialize stackpointer

;StopWDT mov.w #WDTPW|WDTHOLD,&WDTCTL ; Stop watchdog timer

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

call #Init\_UART

Mainloop

call #INCHAR\_UART ; take in first num

call #OUTA\_UART

mov.w R4, R8

call #INCHAR\_UART ; take in second num

call #OUTA\_UART

mov.w R4, R9

mov.w #0x20, R4 ; print a space

call #OUTA\_UART

call #convertTwoCharIntoOne

mov.w R10, R11 ; Result is stored in R11, R8 and R9 and R10 are now free for use.

call #INCHAR\_UART ; take in operation

call #OUTA\_UART

mov.w R4, R12

mov.w #0x20, R4 ; print a space

call #OUTA\_UART

call #INCHAR\_UART ; take in third num

call #OUTA\_UART

mov.w R4, R8

call #INCHAR\_UART ; take in fourth num

call #OUTA\_UART

mov.w R4, R9

mov.w #0x20, R4 ; print a space

call #OUTA\_UART

call #convertTwoCharIntoOne

mov.w R10, R13 ; store second number in R13

mov.w #0x3D, R4 ; print an equals sign

call #OUTA\_UART

mov.w #0x20, R4 ; print a space

call #OUTA\_UART

cmp #0x2B, R12 ; R12 will store the operation

jeq addOp

jmp checkSub

checkSub

cmp #0x2D, R12

jeq subOp

jmp checkMult

checkMult

cmp #0x2A, R12

jeq multOp

addOp

add.w R11, R13

jmp nextLine

subOp

cmp R13, R11

jl negative

jmp notnegative

negative

; print negative

mov.w #0x2D, R4

call #OUTA\_UART

sub.w R11, R13

jmp stopSub

notnegative

sub.w R13, R11 ; R13 contains result

mov.w R11, R13

jmp nextLine

stopSub

multOp

call #getShiftedMult

mov R14, R13

; store result into R13

jmp nextLine

nextLine

; R13 contains result

; print Result

call #printThreeCharFromNum

mov.w #0x0A, R4

call #OUTA\_UART

mov.w #0x0D, R4

call #OUTA\_UART

EXIT jmp Mainloop

getShiftedMult

; R11 = firstnum

; R13 = secnum

mov #0x01, R8 ; x= R8

mov #0x00, R9 ; y = R9

mov #0x00, R10 ; z = zeros = R10

mov #0x00, R14 ; total = R14

mov #0x01, R6 ; firstbit = R6

mov #0x00, R15 ; temp = R5

masterloop

cmp #0x09, R10

jge stopMult

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

mov R13, R6 ; firstbit = secnum

and R8, R6 ; firstbit &= x;

cmp #0x00, R6

jeq setTempZero

jmp tempEqualsFirstNum

tempEqualsFirstNum

mov R11, R15

jmp skip

setTempZero

mov #0x00, R15

skip

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

yloop

cmp R9, R10

jeq stopyloop

rla R15

add #0x01, R9

jmp yloop

stopyloop

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

add R15, R14

rla R8

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

add #0x01, R10

mov #0x01, R6

mov #0x00, R9

jmp masterloop

stopMult ret

; R7 will store the number that needs to be changed from ASCII to actual number

convertFromASCII

cmp.w #0x30, R7

jge checkNextNum

jmp dontCheckNext

checkNextNum

cmp.w #0x40, R7

jl subtractThirty

jmp dontSubtract

subtractThirty

sub.w #0x30, R7

dontSubtract

dontCheckNext

cmp.w #0x41, R7

jge checkOther

jmp exitConvert

checkOther

cmp.w #0x47, R7

jl sub37

jmp dontsub

sub37

sub.w #0x37, R7

dontsub

exitConvert ret

; R7 will store the number that needs to be converted to ASCII

convertToChar

cmp.w #0x0A, R7

jge checkNextNum2

jmp dontCheckNext2

checkNextNum2

cmp.w #0x10, R7 ; 1 more than F

jl subtractThirty2

jmp dontSubtract2

subtractThirty2

add.w #0x37, R7

dontSubtract2

dontCheckNext2

cmp.w #0x00, R7

jge checkOther2

jmp exitConvert2

checkOther2

cmp.w #0x0A, R7

jl sub372

jmp dontsub2

sub372

add.w #0x30, R7

dontsub2

exitConvert2 ret

; R8 will contain first, R9 contains second, R10 will store final value

convertTwoCharIntoOne

mov.w R8, R7

call #convertFromASCII

mov.w R7, R8

mov.w R9, R7

call #convertFromASCII

mov.w R7, R9

; shift R8 four times

; R10 stores the count

mov.w #0x0, R10

rla R8

rla R8

rla R8

rla R8

add.w R8, R10

add.w R9, R10

exitChar ret

printThreeCharFromNum

mov.w R13, R8 ; first

mov.w R13, R9 ; second

mov.w R13, R10 ; third

mov.w R13, R11 ; fourth

and.w #0x0F, R8

and.w #0xF0, R9

and.w #0xF00, R10

and.w #0xF000, R11

rra R9

rra R9

rra R9

rra R9

rra R10

rra R10

rra R10

rra R10

rra R10

rra R10

rra R10

rra R10

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

; now convert them all to ASCI

mov.w R8, R7

call #convertToChar

mov.w R7, R8

mov.w R9, R7

call #convertToChar

mov.w R7, R9

mov.w R10, R7

call #convertToChar

mov.w R7, R10

mov.w R11, R7

call #convertToChar

mov.w R7, R11

; check if third and fourth are zero, R10 and R11

cmp #0x00, R11

jeq zeroIsR11

jmp zeroIsNotR11

zeroIsR11

jmp dontprintR11

zeroIsNotR11

mov.w R11, R4

call #OUTA\_UART

dontprintR11

cmp #0x00, R10

jeq zeroIsR10

jmp zeroIsNotR10

zeroIsR10

jmp dontprintR10

zeroIsNotR10

mov.w R10, R4

call #OUTA\_UART

dontprintR10

mov.w R9, R4

call #OUTA\_UART

mov.w R8, R4

call #OUTA\_UART

exitPrint ret

; LAB UART CODE UNCOMMENT TO TEST IN Lab

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.w &IFG2,R5

and.w #0x02,R5

cmp.w #0x00,R5

jz lpa

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

mov.w R4,&UCA0TXBUF

pop R5

ret

INCHAR\_UART

push R5

lpb mov.w &IFG2,R5

and.w #0x01,R5

cmp.w #0x00,R5

jz lpb

mov.w &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,

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

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

mov.w #0x30,&P2SEL

mov.w #0x00,&UCA0CTL0

mov.w #0x41,&UCA0CTL1

mov.w #0x00,&UCA0BR1

mov.w #0x03,&UCA0BR0

mov.w #0x06,&UCA0MCTL

mov.w #0x00,&UCA0STAT

mov.w #0x40,&UCA0CTL1

mov.w #0x00,&IE2

ret

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

; Interrupt Vectors

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

.sect ".reset" ; MSP430 RESET Vector

.short START ;

.end