CPE 325: Intro to Embedded Computer System

Lab05

Subroutines with Arrays and Program Stack

Submitted by: Dan Otieno

Date of Experiment: 02/10/22

Report Deadline: 02/16/22

Demonstration Deadline: 02/17/22

Introduction

This lab requires calculating the equation Y= MX+C by calling subroutines. The subroutines are called into main, and values are passed through registers in the program stack. There are two methods required, hardware multiplier and binary-shift-add.

Results & Observation

Program 1:

Program Description:

Explain your approach in solving the problem.

For both subroutines, my approach was to define the source codes in separate asm files, and then call them from the main assembly file. An example for the hardware subroutine is already provided in the tutorial, and the method to define and write the instructions to perform that task was modeled from the example in the lab tutorial.

Program Output:

Unable to complete program in time to generate correct output.

Conclusion

Unfortunately, due to time constraints caused by work volume that spilled over from the week prior, I was unable to complete this lab decisively, I felt that the Hardware multiplier code was logical from my perspective, but I couldn't get it to work in time either. I feel left out with this assignment and will revisit with my lab teacher to understand the process better.

Appendix

(Note: These codes are NOT completed, and I submitted them to not miss the deadline, but this is as far as I got).

Table 1: Main program attempt.

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.
                .ref  HW_linear
                .ref    SW_linear
;-------------------------------------------------------------------------------
;                                       USER DATA
;-------------------------------------------------------------------------------
                            .data
;var_C:          .int  2
var_M:          .int  3
arrayX:         .int  1,2,3,4,5,6,7,8,9,1               ; Input: Array X.
arraY:          .int  0,0,0,0,0,0,0,0,0,0
arraySW:        .int  0,0,0,0,0,0,0,0,0,0
;-------------------------------------------------------------------------------
            .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
;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------
main:
                mov.w #var_M, R6
                ;mov.w     #var_C, R7

                push  #10
                push  #arrayX                           ; Push the number of elements.
```

```
                push  #arraY
                call  #HW_linear                ; Call HW_linear.
                inc.w R8
                add.w #8, SP                         ; Collapse the stack.

                push  #10
                push  #arrayX                        ; Push the number of elements.
                push  #arraySW
                call  #SW_linear                ; Call HW_linear.
                inc.w R8
                add.w #8, SP                         ; Collapse the stack.

                jmp       $
;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET
```

Your next code goes here, if any.

Table 02: HW_linear subroutine attempt – unable to complete.

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
            .def    HW_linear
            .text                           ; Assemble into program memory.
HW_linear:
                push  R8
                push  R6
                push  R5
                push  R4

                mov.w      12(SP), R4           ; Retrieve array lengths
```

```
                mov.w       10(SP), R5              ; Array X, input array starting address
                mov.w       8(SP), R8           ; Array Y to store result, starting address

hw_mul:         mov.w       @R5+, &MPY              ; HW multiplier operation, retrieve
element from array X.
                mov.w       R6, &OP2            ; Retrieve integer register.
                nop
                nop
                nop
                mov.w       &RESLO, 0(R8)       ; Multiplication result into array Y.
                dec.w       R4
                jnz   hw_mul

                pop   R4                        ; Collapse stack and
                pop   R5
                pop         R6
                pop         R8

                ret

                .end
```

Table 03: SW_linear attempt.

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
            .def    SW_linear
            .text                           ; Assemble into program memory.
SW_linear:
                    push  R9
                    push  R10
                    push  R11
                    push  R12
                                            ; Retrieve parameters from stack
                    mov.w       16(SP), R4 ; Retrieve array length.
                    mov.w       14(SP), R5 ; ArrayX address.
                    mov.w       12(SP), R7 ; ArraySW address.
```

```
gnext:                clr.w       R12
                      mov.w       @R5+, R9          ;
                      mov.w       @R6+, R10         ;

sw_mul:               bit.w #1, R10
                      jz          sft
                      add.w R9,R12

sft:            rla.w R9
                      rra.w R10
                      dec.w R11
                      jnz         sw_mul
                      bit.w #1, R10
                      jz          lend

lend:           mov.w R12, 0(R7)
                      add.w #2, R7
                      dec.w R4
                      jnz         gnext

                      pop   R12                     ;
                      pop   R11                     ;
                      pop   R10
                      pop   R9

                      ret

                      .end
```