CPE 325: Intro to Embedded Computer System

Lab05

Shift Add Multiplication and Hardware Multiplication

Submitted by: Caleb Keller

Date of Experiment:_____09/18/2022_____

Report Deadline:_____09/21/2022_____

Demonstration Deadline:_____09/22/2022_____

Introduction

Write a brief discussion on what the lab is about. (Use the tutorial and write in your own words. DO NOT copy text).

In this lab, I am writing assembly code to perform two kinds of multiplication methods on 2 *-bit signed integers. The first part of this project will be using the Shift Add method, and the second part will be using the Hardware multiplication method specific to MSP430.

Theory

Write short notes on each topic discussed in lab.

Topic 1: Subroutines

a) Subroutines are used to perform tasks several times on different data values. The sub-routine can be called whenever it is needed within main. A subroutine is executed right after a call instruction. Something like a subroutine, would be a function in C++. Subroutines can be nested meaning a subroutine calls another subroutine.

Topic 2: Passing Parameters

a) Parameters can be passed to a subroutine through the registers, which is straightforward and efficient. They also can be placed in memory locations to be accessed. The stack can be used to pass parameters if there aren't enough registers available.

Results & Observation

Part 1: Shift Add Multiplication

Part 1 Description:

Explain your approach in solving the problem.

To perform the given tasks for this program, I began by writing code to perform multiplication using the Shift Add method. I was not able to display my results, nor check if my code actually performed the method. I followed the lab tutorial and tried my best to attempt it, but was at a loss.

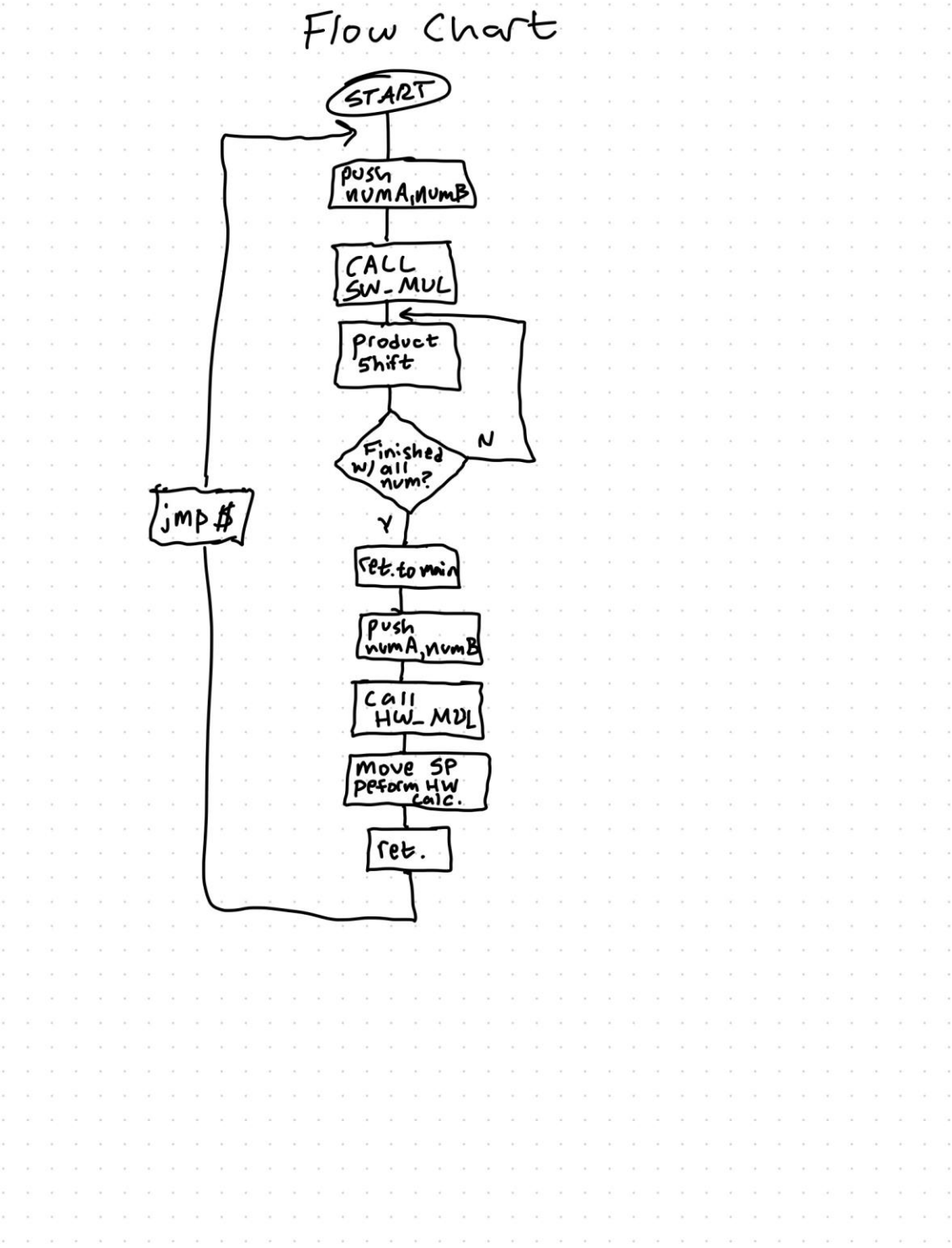
Part 2: Hardware Multiplication

Part 2 Description:

Explain your approach in solving the problem.

In this program, I implemented assembly code to perform the built in hardware multiplication of the integers. Like above in part 1, I was not able to display my results, nor check if they were correct.

Flowchart



Conclusion

While working on this program, I ran into many challenges and issues. I was not able to get my code to generate the values so that I could see what results I was getting. I have came back to this program several times, and was still not able to figure out the issue. I would appreciate in the future, if the programs we are asked to create would have closer relevance to the material that we are actually given. I am frustrated that I was not able to complete this lab correctly, but I have tried my best.

Appendix

Your first code goes here, if any. Make sure you use a 1X1 table for this.

(Note: Make sure the code is readable, have comments. Also reduce spacing between lines to avoid lengthy reports.

Program 1 source code Main.cpp

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;Author: Caleb Keller
;Program: Shift Add and Hardware Multiplication
;Due Date: 09/21/2022
;Description: Calculates the product of 2 8-bit signed integers using subroutines

;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
            .ref    SW_MUL
            .ref    HW_MUL
;-------------------------------------------------------------------------------
            .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.
;-------------------------------------------------------------------------------

numA:         .int   -5
numB:         .int   5

SWres:        .usect "bss", 2
HWres:        .usect "bss", 2

;-------------------------------------------------------------------------------
RESET:  mov.w   #__STACK_END,SP         ; Initialize stack pointer
        mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------
main:
```

```
                    bis.b  #0xFF, &P1DIR    ; configure P1.x as output
                    bis.b  #0xFF, &P2DIR    ; configure P2.x as output
; SW Multiplication
                    push   numA
                    push   numB
                    call   #SW_MUL
; HW Multiplication
                    push   numA
                    push   numB
                    call   #HW_MUL

                    jmp          $


                    bis.w  #LPM4, SR
                    nop
;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack

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

Program 1 Source Code SW_MUL

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;;Author: Caleb Keller
;Program: Shift Add Multiplication
;Due Date: 09/21/2022
;Description: Calculates the product of 2 8-bit signed integers
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
              .def    SW_MUL                    ; Export program entry-point to
;-------------------------------------------------------------------------------
            .text                           ; Assemble into program memory.


SW_MUL:                                 ; save the registers on the stack
                    push   R7     ; save R7, temporal sum
```

```
                    push   R6     ; save R6, array length
                    push   R4     ; save R5, pointer to array
                    push   R8  ; for final routine output
                    clr.w  R7  ; clear R7
                    mov.b  14(SP),      R7 ; retrieve length
                    sxt           R7
                    mov.b  12(SP),      R6     ; retrieve start address
                    sxt           R6
                    mov.w  10(SP), R4 ; get id from the stack
lnext:      inc.w  R9
                    cmp.b  #9, R9
                    jeq           lend
                    bit.w  #1, R6 ; test display id
                    jnz           product
product:    add           R7, R4;
                    rra.w  R6 ;rotate right
                    rla.w  R7 ;rotate left
                    jmp           lnext ; jump back to next

lend:       bit           #1, R6 ; src and des
                    mov.w  R4, 0(R8) ; move r4
                    pop           R7 ; pop off stack
                    pop           R6
                    pop           R4
                    pop           R8
                    sub.w  R7, R4 ; dst+ .not.src +1 -> dst
                    mov.w  R4, &P1OUT ; display result

                    ret ; return to main
                    .end



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

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

Program 1 source code HW_MUL

```
 ;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;;Author: Caleb Keller
```

```
;Program: Hardware Multiplication subroutine
;Due Date: 09/21/2022
;Description: Calculates the product of 2 8-bit signed integers
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
            .def    HW_MUL
;-------------------------------------------------------------------------------
            .text                           ; Assemble into program memory.


HW_MUL:
                    push   R5
                    mov.b  4(SP), R5
                    ;sxt          &MPYS
                    mov.b  8(SP), &MPYS
                    ;sxt          &OP2
                    mov.b  6(SP), &OP2
                    mov.w  RESLO, 0(R5)
                    mov.w  R5, &P2OUT ; display result
                    pop           R5
                    ret           ; return from HW subroutine
                    .end



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

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

Make sure to submit a single pdf file only