Conestoga College – SET

Assignment #6

CodeWarrior 6808 Project

Name	:	Chowon Jung
Student Number	:	8274359
Date of Submission	:	2019-04-07

SEF

(SENG1040)

Table of Contents

Question 1

Table 1.1

Mnemonic / Instruction	Instruction Argument	Addressing Mode	Op-code	Clock Cycle to Execute	Assembled Instruction
PSHA	N/A	IHN	87	2	Push the current accumulator value in order to avoid losing accumulator value the main is holding.
LDA	4, SP	SP1	9EE6	4	Load the value stored in the stack index number 4 to the accumulator.
ADD	#32	IMM	AB	2	Add 32 to the value in the accumulator.
RTS	N/A	INH	81	4	Return to the where this subroutine is called from.
LDA	CELSIUS_ TEMP	DIR	B6	3	Load the value stored in the address of CELSIUS_TEMP into the accumulator.
STA	ANSWER	EXT	C7	4	Store the value current accumulator holding into the memory address of ANSWER.
BRA	mainLoop	REL	20	3	Continue(branch) the program to the mainLoop.

Table 1.2

Event	Total Clock Cycles	Seconds to Execute
EVENT-1: Execute the convCelsius subroutine once	19	0.0011875
EVENT-2: Execute the mainLoop region of code once	39	0.0024375

Question 2

```
/*============================================================================
  The purpose of this program is to perform a "magical" number trick ...

  Tell someone to pick a random number between 1 and 15 and you will get
  them to do some simple math using that number ... at the end of the
trick,
  you will tell them what number they ended up at.  The answer will ALWAYS
be 3!!

  The steps to the trick are:
    1) Get them to pick a random number between 1 and 15 (let's refer to
this as X)
        2) Get them to square X
        3) Now tell them to add X to the answer from (2) above
        4) Now tell them to divide the answer from (3) above by X
        5) Now get them to add 17 to the answer from (4) above
        6) Now get them to subtract X from the answer from (5) above
        7) Finally - have them divide the answer from (6) above by 6

        ANSWER:  3!!!
============================================================================*/


; Include derivative-specific definitions
          INCLUDE 'derivative.inc'
          INCLUDE 'stdio.h'
          INCLUDE 'string.h'


; export symbols
          XDEF _Startup, main
          ; we export both '_Startup' and 'main' as symbols. Either can
          ; be referenced in the linker .prm file or from C/C++ later on

          XREF __SEG_END_SSTACK
; symbol defined by the linker for the end of the stack


; variable/data section
originalSecretNumberPicked:       EQU $80
; Map address of variable "originalSecretNumberPicked" into address $80
intermediateCalc:                 EQU $81
; Map address of variable "intermediateCalc" into address $81
finalAnswer:                      EQU $84
; Map address of variable "finalAnswer" into address $84
```

```
/* ----------------------------------------------------------- */
/* The following three functions are simple supporting      */
/* mathematical functions - for SQUARING a number, ADDING   */
/* two numbers together and DIVIDING one number by another  */
/* ----------------------------------------------------------- */
squareIt:
                     PSHA
; Preserve the A register values upon being called
                     LDX 4, SP
; Load the int numToSquare into the accumulator
                     MUL
; Multiply A value by X value
                     STA 4, SP
; Store the result into the stack index number 4
                     PULA
; Pop the saved A value off the stack
                     RTS
; Return to where this subroutine was called from

addThem:
                     PSHA
; Preserve the A register values upon being called
                     LDA 4, SP
; Load the int numOne into the accumulator
                     ADD 5, SP
; Add the int numTwo into the accumulator
                     STA 4, SP
; Store the result into the stack index number 4
                     PULA
; Pop the saved A value off the stack
                     RTS
; Return to where this subroutine was called from

divideNumOneByNumTwo:
                     PSHH
; Preserve the H register values upon being called
                     PSHX
; Preserve the X register values upon being called
                     PSHA
; Preserve the A register values upon being called
                     LDX 6, SP
; Load the int numOne into the accumulator
                     CLRH
; Clear out the H register
                     LDA 7, SP
; Load the int numTwo into the accumulator
                     DIV
; Divide A value by X value
                     STA 6, SP
; Store the result into the stack index number 6
                     PULA
; Pop the saved A value off the stack
                     PULX
; Pop the saved X value off the stack
                     PULH
; Pop the saved H value off the stack
                     RTS
```

```
; Return to where this subroutine was called from


main:
_Startup:
          LDHX   #__SEG_END_SSTACK
; initialize the stack pointer
          TXS
          CLI
; enable interrupts

mainLoop:
                    LDA    #08
; Load the constant value of 8 decimal number into accumulator
                    STA originalSecretNumberPicked
; Store the data in the accumulator into memory location of variable
"originalSecretNumberPicked"
                    LDA #00
; Load the constant value of 0 decimal number into accumulator
                    STA intermediateCalc
; Store the data in the accumulator into memory location of variable
"intermediateCalc"
                    LDA #00
; Load the constant value of 0 decimal number into accumulator
                    STA finalAnswer
; Store the data in the accumulator into memory location of variable
"finalAnswer"


                    LDA originalSecretNumberPicked
; Load the data in the memory location of variable
"originalSecretNumberPicked" into accumulator
                    PSHA
; Push the data on the accumulator into the stack where the stack pointer
is pointing
                    JSR squareIt
; Jump to the subroutine "squareIt"
                    PULA
; Pop the saved A value off the stack
                    STA intermediateCalc
; Store the data in the accumulator into memory location of variable
"intermediateCalc"


                    LDA intermediateCalc
; Load the data in the memory location of variable "intermediateCalc" into
accumulator
                    PSHA
; Push the data on the accumulator into the stack where the stack pointer
is pointing
                    LDA originalSecretNumberPicked
; Load the data in the memory location of variable
"originalSecretNumberPicked" into accumulator
                    PSHA
; Push the data on the accumulator into the stack where the stack pointer
is pointing
                    JSR addThem
```

```
; Jump to the subroutine "addThem"
                    PULA
; Pop the saved A value off the stack
                    AIS #1
; Clean up the stack 1 byte remaining
                    STA intermediateCalc
; Store the data in the accumulator into memory location of variable
"intermediateCalc"

       ;
                    LDA intermediateCalc
; Load the data in the memory location of variable "intermediateCalc" into
accumulator
                    PSHA
; Push the data on the accumulator into the stack where the stack pointer
is pointing
                    LDA originalSecretNumberPicked
; Load the data in the memory location of variable
"originalSecretNumberPicked" into accumulator
                    PSHA
; Push the data on the accumulator into the stack where the stack pointer
is pointing
                    JSR divideNumOneByNumTwo
; Jump to the subroutine "divideNumOneByNumTwo"
                    PULA
; Pop the saved A value off the stack
                    AIS #1
; Clean up the stack 1 byte remaining
                    STA intermediateCalc
; Store the data in the accumulator into memory location of variable
"intermediateCalc"


                    LDA intermediateCalc
; Load the data in the memory location of variable "intermediateCalc" into
accumulator
                    PSHA
; Push the data on the accumulator into the stack where the stack pointer
is pointing
                    LDA #17
; Load the constant value of 17 decimal number into accumulator
                    PSHA
; Push the data on the accumulator into the stack where the stack pointer
is pointing
                    JSR addThem
; Jump to the subroutine "addThem"
                    PULA
; Pop the saved A value off the stack
                    AIS #1
; Clean up the stack 1 byte remaining
                    STA intermediateCalc
; Store the data in the accumulator into memory location of variable
"intermediateCalc"
```

```
                    LDA intermediateCalc
; Load the data in the memory location of variable "intermediateCalc" into
accumulator
                    PSHA
; Push the data on the accumulator into the stack where the stack pointer
is pointing
                    LDA originalSecretNumberPicked
; Load the data in the memory location of variable
"originalSecretNumberPicked" into accumulator
                    NEGA
; Negate the value on the accumulator
                    PSHA
; Push the data on the accumulator into the stack where the stack pointer
is pointing
                    JSR addThem
; Jump to the subroutine "addThem"
                    PULA
; Pop the saved A value off the stack
                    AIS #1
; Clean up the stack 1 byte remaining
                    STA intermediateCalc
; Store the data in the accumulator into memory location of variable
"intermediateCalc"


                    LDA intermediateCalc
; Load the data in the memory location of variable "intermediateCalc" into
accumulator
                    PSHA
; Push the data on the accumulator into the stack where the stack pointer
is pointing
                    LDA #06
; Load the constant value of 6 decimal number into accumulator
                    PSHA
; Push the data on the accumulator into the stack where the stack pointer
is pointing
                    JSR divideNumOneByNumTwo
; Jump to the subroutine "divideNumOneByNumTwo"
                    PULA
; Pop the saved A value off the stack
                    AIS #1
; Clean up the stack 1 byte remaining
                    STA finalAnswer
; Store the data in the accumulator into memory location of variable
"finalAnswer"


          BRA   mainLoop
; Branch to the mainLoop
```