SEng 1040

SENG 1040

Assignment #5

Due date: March 26, 2019

Student: Lev Potomkin

Question 1

Table 1.1

Instruction	Addressing mode	# of clock cycles
CLRA	Inherent	1
STA $03	Direct	3
LDA #$FF	Immediate	2
STA $01	Direct	3
STA $43	Direct	3
LDA $00	Direct	3
NSA	Inherent	3
STA $40	Direct	3
JMP MainLoop	Extended	3

Table 1.2

Event	Total # of clock cycles	Elapsed wall-clock time (s)
1. Execute IOSetup once	1 + 3 + 2 + 3 + 3 = **12**	12*0.000000125 = **0.0000015**
2. Execute MainLoop once	3 + 3 + 3 + 3 = **12**	12*0.000000125 = **0.0000015**
3. Execute 10 iterations of the program (from the start)	12 + 10*12 = **132**	132*0.000000125 = **0.000018**

*Note: assuming 8MHz frequency, as found in the manual

Question 2

```
; variable/data section
firstOperand:  EQU $80
secondOperand: EQU $81
sum:           EQU $84
difference:    EQU $86

mainLoop:
    LDA #18           ; put 18 decimal in A
    STA firstOperand  ; store it in memory
    LDA #8            ; put 8 decimal in A
    STA secondOperand ; store it in memory

    ; this part isn't really necessary
    CLRA              ; put 0 in A
    STA sum           ; store it in sum
    STA difference    ; also store it in difference

    LDA firstOperand  ; load the first operand
    PSHA              ; push it onto the stack
    LDA secondOperand ; load the second operand
    PSHA              ; push it onto the stack

    JSR calculateSum  ; call the sum subroutine
    PULA              ; pop the result
    AIS #1            ; clean up the stack
    STA sum           ; store the result in sum

    LDA firstOperand  ; load the first operand
    PSHA              ; push it onto the stack
    LDA secondOperand ; load the second operand
    PSHA              ; push it onto the stack

    JSR calculateDifference  ; call the difference subroutine
    PULA                     ; pop the result
    AIS #1                   ; clean up the stack
    STA difference           ; store the result in difference

theEnd:
      BRA theEnd        ; suspend execution


calculateSum:
    PSHA       ; preserve A
    LDA 5, SP  ; load the first argument
    ADD 4, SP  ; add the second argument
    STA 4, SP  ; store the result
    PULA       ; pop A (the one that was before the call)
    RTS        ; return

calculateDifference:
    PSHA              ; preserve A

    LDA 5, SP         ; load the first argument
    PSHA              ; push it onto the stack

    LDA 5, SP         ; load the second argument (index 5 because we pushed
one more before)
    NEGA              ; negate it
    PSHA              ; push it onto the stack

    JSR calculateSum  ; call the sum subroutine
    PULA              ; pop the result
    AIS #1            ; clean up the stack

    STA 4, SP         ; store the result
    PULA              ; pop A (the one that was before the call)
    RTS               ; return
```