SENG 1040 – CAML
Assignment #6
Due Date: April 9, 2019
Student: Lev Potomkin (alone)

Question 1

Table 1.1

Mnemonic	Argument	Addressing	Opcode	# of cycles	Assembled
PSHA		Inherent	87	2	87
LDA	4, SP	Indexed	9E E6	4	9E E6 04
ASLA		Inherent	48	1	48
ADD	#32	Immediate	AB	2	AB 20
STA	4, SP	Indexed	9E E7	4	9E E7 04
PULA		Inherent	86	2	86
RTS		Inherent	81	4	81
LDA	$80	Direct	B6	3	B6 80
PSHA		Inherent	87	2	87
CLRA		Inherent	4F	1	4F
JSR	$1000	Extended	CD	5	CD 10 00
PULA		Inherent	86	2	86
STA	$100	Extended	C7	4	C7 01 00
BRA	$2000	Relative	20	3	20 20 00

Table 1.2

Event	# of clock cycles	Wall-clock time (s)
convCelsius	19	19*0.0000625=**0.0011875**
mainLoop	39	39*0.0000625=**0.0024375**

Question 2

```
; variables
secretNumber: EQU $80
finalAnswer: EQU $84

; this is the main function that performs all calculations
program:
    LDA #8              ; load A with 8 decimal
    STA secretNumber    ; store it at $80
    PSHA                ; push it onto the stack
    JSR squareIt        ; call the squareIt subroutine
    PSHA                ; at this point we have 64 on the stack, push 8 again
    JSR addThem         ; call the addThem subroutine on 64 and 8
    AIS #1              ; pop an extra byte from the stack
    PSHA                ; at this point we have 72 on the stack, push 8 again
    JSR divideIt        ; call the divideIt subroutine on 72 and 8
    AIS #1              ; pop an extra byte from the stack
    LDA #17             ; load A with 17 decimal
    PSHA                ; at this point we have 9 on the stack, push 17
    JSR addThem         ; call the addThem subroutine on 9 and 17
    AIS #1              ; pop an extra byte from the stack
    LDA secretNumber    ; load A with 8 again
    NEGA                ; perform the 2's complement
    PSHA                ; push it onto the stack
    JSR addThem         ; call the addThem subroutine on -8 and 26
    AIS #1              ; pop an extra byte from the stack
    LDA #6              ; load A with 6 decimal
    PSHA                ; at this point we have 18 on the stack, push 6
    JSR divideIt        ; call the divideIt subroutine on 18 and 6
    AIS #1              ; pop an extra byte from the stack
    PULA                ; load A with whatever is on top of the stack and pop it
    STA finalAnswer     ; store A at $84

endOfProgram:
    BRA endOfProgram    ; stay here forever
```

```
; this function takes one byte, squares it and puts it onto the stack
squareIt:
    PSHA            ; preserve A
    PSHX            ; preserve X
    LDA 5, SP       ; load A from the stack (the argument)
    LDX 5, SP       ; load X from the stack (same as A)
    MUL             ; multiply X and A, store result in X:A
                    ; that's why our number should be less than 16
    STA 5, SP       ; store the result on the stack
    PULX            ; restore X
    PULA            ; restore A
    RTS             ; return

; this function takes 2 bytes, adds them and puts result onto the stack
addThem:
    PSHA            ; preserve A
    LDA 5, SP       ; load A with the first argument
    ADD 4, SP       ; add the second argument to A
    STA 5, SP       ; store the result on the stack
    PULA            ; restore A
    RTS             ; return

; this function takes 2 bytes, divides first one by the second one
; and puts result onto the stack
divideIt:
    PSHA            ; preserve A
    PSHH            ; preserve H
    PSHX            ; preserve X
    LDA 7, SP       ; load A from the stack (first argument0
    CLRH            ; set H to zero
    LDX 6, SP       ; load X from the stack (second argument)
    DIV             ; divide A by X and store result in A
    STA 7, SP       ; store the result on the stack
    PULX            ; restore X
    PULH            ; restore H
    PULA            ; restore A
    RTS             ; return
```