

Part I – Programming

You will create a program that will compute the arithmetic result of addition, subtraction, multiplication, and division. The program is to be broken into two parts: arithmetic practice, and temperature conversion.

In the first part of the program you are to prompt the user for two numbers. With these numbers, you will: compute the addition and display the result, subtract these two numbers and display the result, multiply these two numbers and display the result, and finally divide these two numbers and display the result. You should display the result the preceding order and each result should be labeled with some output indicating what type of arithmetic operation was performed.

In the second part you will again prompt the user for a Fahrenheit temperature. With this temperature acquired, you will convert it to its Celsius equivalent and display the result (indicate in the output that the result is the Celsius equivalent). The conversion formula from Fahrenheit to Celsius is as follows:

$$\text{Celsius} = (\text{Fahrenheit} - 32) * 5/9$$

You then need to compute the Kelvin temperature equivalent and display the result to standard out (indicate in the output that the result is the Kelvin equivalent). The conversion formula from Fahrenheit to Celsius is as follows:

$$\text{Kelvin} = (\text{Fahrenheit} - 32) * 5/9 + 273$$

You may use any functions from the “util.lib,” file. You will need to use “PutDec,” to display numbers to standard out and the function “GetDec,” to acquire input from the user. You can use the aforementioned functions by adding an external definition to your program. You do with the syntax, `EXTERN <FUNCTION NAME>:<QUALIFIED TYPE>`. For example for the function “GetDec,” you would add the extern line in a location that is above the elaboration of all functions that use that externally defined function (you only need have declared the external function once). The `<QUALIFIED TYPE>` in this case would be the type of pointers that are used for this function and should be “NEAR.” The following demonstrates

```
EXTERN GetDec:near
```

The external declaration of the function name is case sensitive and usually follows the C language standards unless otherwise specified.

The input will be positive numbers as well as negative numbers, therefore, use the appropriate arithmetic operators.

Grade distribution:

Part 1	
Addition	10%
Subtraction	10%
Multiplication	10%
Division	10%
Fahrenheit Conversion	
Celsius	10%
Kelvin	10%
Part 2	10%
Following Instructions	10%
Comments	
Program Description	5%
Code Comments	5%
Misc./Efficiency	10%

Part II – Questions and Answers

1) Run your program 4 times and fill in the table with the given input

Input		Output			
1 st num	2 nd num	Add	Subtract	Multiply	Divide
10	5	15	5	50	2
10	10	20	0	100	1
1	2	3	-1	2	0
-1	1	0	-2	-1	-1
0	0	0	0	0	Undefined

Input		Output	
Fahrenheit	Celsius	Kelvin	
32	0	273	
-32	-35	238	
-1	-18	255	
65535	-17/-18	256/255	
0	-17	256	

2) Convert the following numbers from one number system to the next (<#>_b means # in base number system b, i.e. 10₂ is 10 in the binary number system or 2 in the decimal number system)

- 32₁₀ to binary = 2⁵ = 0010 0000₂
- 10100101₂ to hexadecimal = 0A5₁₆
- 10100101₂ to octal = 245₈
- FFE5₁₆ to binary = 1111 1111 1110 0101₂
- 43₁₀ to base 7 = 61₇

```
;;P1-1234.asm
title Farenheit
```

```
comment ~ *****
```

```
Name:          Bubba
SSN:           123-45-6789
Section:       09422
Date:          2/17/2004
Assignment:    Program 1
```

Description:

This program is broken into two parts.
The first part demonstrates arithmetic operations:
addition, subtraction, multipliation and division.
Two numbers are taken from the input and the arithmetic
operations are performed on them.

The second part of the program requests a Farenheit
temperature value from the user and displays the
the according Celsius and Kelvin temperature values.

The formula to convert Farenheit to Celsius is
 $C = (F - 32) * 5/9$

The formula to convert Celsious to Kelvin is
 $K = C + 273$

Input:

First number - the first number which additions and
multiplications will be made to and
from which subtractions and division will be
made from
Second number - the number which will be added to or
multiplied to the first number and the number
that will be subtracted from the first and
the number that will divide the first
Farenheit - the temperature to convert to Celsius and
Kelvin

Output:

Stdout - the result of arithmetic operations and the
the result of temperature conversions

```
~ *****
```

```
.model small
.586
.stack 100h
.nolist
```

```
;*****
;Includes
;*****
include bios.inc
includelib util.lib
```

```
.list
.data
```

```
;*****
;Defintion declarations
;*****
CR equ 0Dh
LF equ 0Ah
celsiusconst equ 32
kelvinconst equ 273
```

```
;*****
;Variables and constant declarations
;*****
```

```
crlf db CR,LF
lenCrLf equ lengthof crlf ;get length of crlf
mess1 db "Enter first number:"
len1 equ lengthof mess1 ;get length of message1
mess2 db "Enter second number:"
len2 equ lengthof mess2 ;get length of message2
```

```

mess3 db "Result of addition:"
len3 equ lengthof mess3 ;get length of message3
mess4 db "Result of subtraction:"
len4 equ lengthof mess4 ;get length of message4
mess5 db "Result of multiplication:"
len5 equ lengthof mess5 ;get length of message5
mess6 db "Result of division:"
len6 equ lengthof mess6 ;get length of message6
mess7 db "Enter farenheiht measurement:"
len7 equ lengthof mess7 ;get length of message7
mess8 db "Celsius equiv:"
len8 equ lengthof mess8 ;get length of message8
mess9 db "Kelvin equiv:"
len9 equ lengthof mess9 ;get length of message8

first sword ?
second sword ?

.code ;begin the code segment

;*****
;Function prototypes
;*****
extern GetDec:near, PutDec:near
displaymsg proto near c m:word,l:byte

main proc
.startup ;start the main procedure
@Cls ;clear the screen
@SetCsrPos 0, 0 ;set the cursor position

;PART I
;*****
;Get the first number
;*****
invoke displaymsg, addr mess1, len1 ;display mess1
call getdec ;get first number from stdin
mov first, ax ;store the number to variable first

;*****
;Get the second number
;*****
invoke displaymsg, addr mess2, len2 ;display mess2
call getdec ;get second number from stdin
mov second, ax ;store the number to variable second
invoke displaymsg, addr crlf, lenCrLf ;display crlf

;*****
;Display first+second
;*****
invoke displaymsg, addr mess3, len3 ;display mess3
mov ax, first ;move first into AX register
mov bx, second ;move second into BX register
add ax, bx ;add AX and BX, store in AX
call putdec ;display AX register to stdout (addition)
invoke displaymsg, addr crlf, lenCrLf ;display crlf

;*****
;Display first-second
;*****
invoke displaymsg, addr mess4, len4 ;display mess4
mov ax, first ;move first into AX register
mov bx, second ;move second into BX register
sub ax, bx ;subtract BX from AX, store in AX
call putdec ;display AX register to stdout (subtraction)
invoke displaymsg, addr crlf, lenCrLf ;display crlf

;*****
;Display first*second
;*****
invoke displaymsg, addr mess5, len5 ;display mess5

```

```

mov dx, 0 ;clear DX
mov ax, first ;move first into AX register
mov bx, second ;move second into BX register
imul bx ;multiply AX and BX, store in AX
call putdec ;display AX register to stdout (multiplication)
invoke displaymesg, addr crlf, lenCrLf ;display crlf

;*****
;Display first/second
;*****
invoke displaymesg, addr mess6, len6 ;display mess6
mov ax, first ;move first into AX register
mov bx, second ;move second into BX register
idiv bx ;divide AX by BX, store in AX
call putdec ;display AX register to stdout (division)
invoke displaymesg, addr crlf, lenCrLf ;display crlf
invoke displaymesg, addr crlf, lenCrLf ;display crlf

;PART II
;*****
;Get the fahrenheit number
;*****
invoke displaymesg, addr mess7, len7 ;display mess7
call getdec ;get fahrenheit number from stdin
sub ax, celsiusconst ;subtract celsius constant from AX
mov dx, 0 ;clear DX
mov bx, 5 ;move 5 into BX
imul bx ;signed multiply AX by BX
mov bx, 9 ;move 9 into BX
idiv bx ;signed divide AX by BX
invoke displaymesg, addr mess8, len8 ;display mess8
call putdec ;display AX register to stdout (celsius)
invoke displaymesg, addr crlf, lenCrLf ;display crlf
add ax, kelvinconst ;add kelvinconst to AX
invoke displaymesg, addr mess9, len9 ;display mess9
call putdec ;display AX register to stdout (kelvin)
invoke displaymesg, addr crlf, lenCrLf ;display crlf
.exit ;end the main procedure

main endp

;*****
;Function: displaymesg
;Description:
; This function takes a pointer to a string
; as input and the length of the string and
; prints the result to stdout. It does this
; by using the OS int 21h and function 040h.
; BX holds the device (1), cl holds the
; length of string, dx holds the address of
; the string to be displayed. Flags and
; the 32-bit registers are pushed and popped
; of the stack.
;Input:
; m - a pointer to the string which is to be
; displayed to stdout
; l - an short integer indicating the length
; of the string to display
;Output:
; None
;Return:
; None
;*****
displaymesg proc near c uses ax bx cx dx, m:word,l:byte
    pushf ;push the flags onto the stack
    pushad ;save all registers on the stack
    mov ah, 040h ;move function number 040h into ah
    mov bx, 1 ;move device number into bx
    mov cl, l ;move the length of the message into cx
    shl ch, 8 ;clear out high part of cx
    mov dx, m ;move the message address into dx
    int 21h ;request interrupt service from OS

```

```
    popad
    popf
    ret
displaymsg endp
end
```

```
;pop all of the registers off of the stack
;pop and restore the flags off of the stack
;return flow control the caller
```