

**Western Michigan university**

Electrical and Computer Engineering department

**ECE-2510-560-Intro to microprocessor**

**LAB 7**

**Subroutines, Local Variables, and**

**Implementing Pseudocode**

Kaushik Mojumder and Sutopa Suhrid

Name of Laboratory Instructor- Husam Beitello  
Day/Time of your lab section-2.30 PM-5.10 PM, Thursday

Date Laboratory was Performed-October 27,2021

Table of Contents

**PROCEDURES3**

Task 13

Task 212

**Analysis and Conclusion………………………………………………………………17**

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

1. **Procedures:**
   1. ***Task 1:***

For this task, we made two different arrays. The first array contains 6 words, and the second array contains only 2 words. We made a program such that the original array is overwritten by the following equation:

Array =

First, we created a new project and renamed to lab7task1. Then we cleared the sample Fibonacci counter program and all the variables.

We compiled the code without any error and the no error dialogue also appeared.

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

Graphical user interface, application

Description automatically generated

Figure: task 1 compile message

Under Ram section, we declared the array:

ORG $1100; starting address is m[1100]

arr1: dc.w 1,3,5,7,9,11; m[1100] = 00,

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

m[1101] = 01, m[1102] = 00, m[1103] = 03, m[1104] = 00, m [1105] = 05, m[1106] = 00,

m[1107] = 07, m[1108] = 00, m[1109] = 09, m[110A] = 00, m[110B] =0B

arr1len: dc.b 6; m[110C] = 06

arr2: dc.w 15,100; m[110D] = 00, m[110E] = F, m[110F] = 00, m[1200] =64

arr2len: dc.b 2; m[1201] = 2

Under endless loop we added the following code:

LDX #arr1; X = #$1100

LDAB arr1len; B = 6

JSR ArrayPolynomial2

NOP

LDX #arr2;X = #$110D

LDAB arr2len; B = 2

JSR ArrayPolynomial2

NOP

BRA EndlessLoop

ArrayPolynomial2:

Loop4: STAB $1200; m[1200] = B = 6

LDD 0,X; D = m[0+X] = m[1100] = 01

Std $1210; m[1210] = 00, m[1211] = 01

tfr D,Y; Y = D = 01

emul; D x Y = Y:D

Addd $1210; D = D + m[1210]

Addd #10; D = D + #$A

STD 2,X+; D = m[1100], X = m[1100+2] = m[1102]

BRA Loop3; Branch to Loop3

Loop3:

LDAB $1200; B = m[1200] =6

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

DBNE B, Loop4; B = B-1 = 6-1 =5. Branch to loop4 if B>0

RTS; jump back to the point where subroutine started

The overall code looks like this:

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

A picture containing text

Description automatically generated

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

Figure: task 1 code

The DRAGON12 Plus-2 was reset and the bootloader was uploaded. Then the program was debugged. We set a breakpoint at NOP so that we can see the result at the array. The initial values of the arrays will be replaced with the result of the given equation.

Graphical user interface, application

Description automatically generated

Figure: data window for the array

The values of the array can be seen in the data window before running the code.

Text

Description automatically generated

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

Figure: memory view of the array values

The memory locations are also filled with the values of the array and arraylen

Graphical user interface, application

Description automatically generated

Figure: breakpoint at NOP1

We ran the full code for the subroutine to find the result the equation for array1 values and set a breakpoint at NOP1. Now we can see that the result overwrote the original array1 values.

By hand calculation:

Array =

Array = +1+A = C

M[1100] = 00, m[1101] =0C which is verified in the memory window

Array = +3+A = 16

M[1102] = 00, m[1103] = 16 which is verified in the memory window

Array = +5+A = 28

M[1104] = 00, m[1105] =28 which is verified in the memory window

Array = +7+A = 42

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

M[1106] =00, m[1107]=28 which is verified in the memory window

Array =

M[1108]= 00, m[1109] = 64 which is verified in the memory window

Array = +B+A =8E

M[110A] = 00, m[110B] = 8E which is verified in the memory window

Graphical user interface, application

Description automatically generated

Figure: breakpoint at NOP2

We ran the full code for the subroutine to find the result the equation for array1 values and set a breakpoint at NOP1. Now we can see that the result overwrote the original array2 values.

By hand calculation:

Array =

Array = +F+A = FA

M[110D] =00, M[110E] = FA which is verified at the memory address

Array = +64+A = 277E

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

M[110F] = 27, m [1200] 7E which is verified at the memory address

The subroutine code:

ArrayPolynomial2:

Loop4:

STAB $1200; m[1200] = B = 6

LDD 0,X; D = m[0+X] = m[1100] = 01

Std $1210; m[1210] = 00, m[1211] = 01

tfr D,Y; Y = D = 01

emul; D x Y = Y:D

Addd $1210; D = D + m[1210]

Addd #10; D = D + #$A

STD 2,X+; D = m[1100], X = m[1100+2] = m[1102]

BRA Loop3; Branch to Loop3

Loop3:

LDAB $1200; B = m[1200] =6

DBNE B, Loop4; B = B-1 = 6-1 =5. Branch to loop4 if B>0

RTS; jump back to the point where subroutine started

First, we loaded the value of memory address pointed by X at register D. So, D has the word of array. We stored D at $1210. Then we transferred D to register Y to calculate the square of word. The lower bits are stored at D. We added D with the value of memory location $1210. This computed array^2 + array. Then we added #10. This computed array^2 + array + A. We loaded the register B with array length. Then each time a loop4 was done, we decrement register B by 1 and store it at m[1200]. Then, register B is loaded with m[1200] again. We went back to the main program once register B was loaded with 0.

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

* 1. ***Task 2:***

For task 2 we calculated the GCD (greatest common divisor) for 1071(#$42F) and 1029(#$405). First, we created a new project and renamed to lab7task2. Then we cleared the sample Fibonacci counter program and all the variables.

We compiled the code without any error and the no error dialogue also appeared.

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

Graphical user interface, text, application

Description automatically generated

Figure: task 2 compile message

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

We added the following code under Endlessloop:

LDD #1071; D = #$42F

PSHD; SP= SP-2= 1100-2 = 10FE, m[10FE] = 04, m[10FF] = 2F

LDD #1029; D = #$405

PSHD; SP = SP-2= 10FE-2 = 10FC; m[10FC] =04, m[10FD] = 05

JSR GCD; jump to subroutine GCD

PULD

PULD

NOP

BRA EndlessLoop

GCD:

LEAS -2,SP; allocate K

LDX 4,SP ; load j into X

Loop2:

Beq Loop2Done; branches to Loop2Done if z flag = 1

LDD 6,SP;load I into D

idiv; D/X , X = quotient, D = remainder

STD 0,SP; store D at variable K

LDD 4,SP; Load D with J

STD 6,SP; store D at I. so, I = J

LDX 0,SP; load X with K

STX 4,SP; store X with J. so, J = K

BRA Loop2; Branch to Loop2

Loop2Done:

LEAS 2,SP;deallocate k

RTS; return to subroutine

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

The overall code looks like this:

Text

Description automatically generated with medium confidence

Figure: task 2 code

The DRAGON12 Plus-2 was reset and the bootloader was uploaded. Then the program was debugged. We set a breakpoint at NOP.

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

Graphical user interface, application, Word

Description automatically generated

Figure: GCD of 1071 and 1029

The GCD of 1071 and 1029 is 21. From the value calculated in lab, D = $15

$15 = 21 in decimal. So the GCD matched the results.

The subroutine code is added below:

GCD:

LEAS -2,SP; allocate K

LDX 4,SP ; load j into X

Loop2:

Beq Loop2Done; branches to Loop2Done if z flag = 1

LDD 6,SP;load I into D

idiv; D/X , X = quotient, D = remainder

STD 0,SP; store D at variable K

LDD 4,SP; Load D with J

STD 6,SP; store D at I. so, I = J

LDX 0,SP; load X with K

STX 4,SP; store X with J. so, J = K

BRA Loop2; Branch to Loop2

Lab7: Subroutines, Local Variables, and Implementing Pseudocode R- 2.30

Loop2Done:

LEAS 2,SP;deallocate k

RTS; return to subroutine

When we used LEAS -2,SP it allocated the local variable K. As a result, K was allocated at 0, SP.

When we used LEAS 2, SP it deallocated the local variable K.

To find the GCD the remainder must be 0. It will set the Z flag to 1 when the remainder becomes 0. We kept looping into loop2 until Z was set to 1. Each time I was changed to J and J was changed to K. So, the value of register D and X was changed. We kept dividing D by X. When Z =1 we moved to Loop2Done.

1. **Analysis:**

For task 1, we declared two arrays. We also declared length of two arrays so that we can iterate the loop for a specific number of times. We calculated the following equation:

Array =

We used register D and Y to perform the square of the word. Register D was used to perform the remaining operations. We also flooded the original array values with new calculations. The hand analysis and the lab results were verified.

For task 2, we made a while loop to find out the greatest common divisor for 1071

and 1029. Inside the loop, we continued the division until the remainder becomes 0. The BEQ instruction checked the Z flag of CCR. When Z =1 we obtained the GCD.