**Lab Manual – Subroutines**

**Important Note: Make stack on your paper and then read, write, push and pop data accordingly.** **If you are using registers in a function make sure you save and restore their values.**

1. Consider the following piece of C++ code fragments. Write their equivalent assembly language programs. Write complete code to call the function and save the returned value. Test your functions on different parameters on AFD.

|  |
| --- |
| int goose()  {  return -4;  } |
| int sky (int a, int b)  {  Int localVar = a-b;  return localVar;  } |
| int sheep(int c)  { if(c < 0)  return 1;  else  return 0; } |
| int duck(int a)  { if(sheep(a))  return -a;  else  return a; } |

1. Write a subroutine that takes as parameter the address of array and the number of elements in it through stack and returns the sum of elements through stack as well. Test your program on two different function calls i.e. for two different arrays.

**Practice Problems**

1. Consider a return value, 4 parameters, 2 local variables (of subroutine) and 4 register copies are placed on stack.
2. How can we access all parameters in function/subroutine?
3. How can we place return value of function?
4. How can we access local stack variables of the function/sub routine?
5. How to empty stack before & after leaving subroutine?
6. How can we pass parameters and retrieve return value in Caller?
7. **[Multiple Function Calls]** Go through the code given below. Read the comments carefully to understand the code. Note the exact configuration of stack and required registers (content of memory, SP and IP) for all the calls. Verify your answer with AFD.

|  |
| --- |
| ; multiple subroutines call and ret instruction - 4 function calls  [ORG 0x0100]  jmp start  a: db 0xA  b: db 0xB  c: db 0xC  min: db 0xF  max: db 0x0  total: db 0x0  avg: db 0x0  ans: db 0x0  ;-----------------------------------------------------  ; FindMin finds the minimum of a,b and c  ;-----------------------------------------------------  FindMin: mov al, [a]  cmp al, [b]  ja bIsSmaller ; b < a  cmp al, [c] ; either a or c is min  ja cIsSmallest ; a>c  mov dl, al  jmp return  bIsSmaller: mov al, [c]  cmp al, [b]  ja bIsSmallest ; b < c  mov dl, [c] ; c is smallest  jmp return  bIsSmallest: mov dl, [b]  jmp return  cIsSmallest: mov dl, [c] ;dl = c  return: mov [min], dl  ret  ;-----------------------------------------------------  ; FindMax finds the maximum of a,b and c  ;-----------------------------------------------------  FindMax: mov al, [a]  cmp al, [b]  jb bIsGreater ; b > a  cmp al, [c] ; either a or c is max  jb cIsGreatest ; c > a  mov dl, al  jmp returnFindMax  bIsGreater: mov al, [c]  cmp al, [b]  jb bIsGreatest ; b < c  mov dl, [c] ; c is greatest  jmp returnFindMax  bIsGreatest: mov dl, [b]  jmp returnFindMax  cIsGreatest: mov dl, [c] ;dl = c  returnFindMax: mov [max], dl    ret  ;-----------------------------------------------------  ; FindTotal finds the sum of a,b and c  ;-----------------------------------------------------  FindTotal: mov al, 0  add al, [a]  add al, [b]  add al, [c]  mov [total], al  ret  ;-----------------------------------------------------  ; FindAvg calculates the Average of a,b and c  ; See 8088 Manual for the details of DIV Instruction  ;-----------------------------------------------------  FindAvg: mov al, [total]  mov cl, 3  div cl ;al = al/cl  mov [avg], al  ret  ;-----------------------------------------------------  start: call FindMin  call FindMax  call FindTotal  call FindAvg  MOV AX, 0x4C00 ; Terminate Program  INT 0x21 |

1. **[Nested Function Calls]** Go through the code given below. Read the comments carefully to understand the code. Note the exact configuration of stack and required registers (content of memory, SP and IP) for all the calls. Verify your answer with AFD.

|  |
| --- |
| ; Nested function call  [ORG 0x0100]  jmp start  a: db 0xA  b: db 0xB  c: db 0xC  min: db 0xF  max: db 0x0  total: db 0x0  avg: db 0x0  ans: db 0x0  ;-----------------------------------------------------  ; FindMin finds the minimum of a,b and c  ;-----------------------------------------------------  FindMin: mov al, [a]  cmp al, [b]  ja bIsSmaller ; b < a  cmp al, [c] ; either a or c is min  ja cIsSmallest ; a>c  mov dl, al  jmp return  bIsSmaller: mov al, [c]  cmp al, [b]  ja bIsSmallest ; b < c  mov dl, [c] ; c is smallest  jmp return  bIsSmallest: mov dl, [b]  jmp return  cIsSmallest: mov dl, [c] ;dl = c  return: mov [min], dl  ret  ;-----------------------------------------------------  ; FindMax finds the maximum of a,b and c  ;-----------------------------------------------------  FindMax: mov al, [a]  cmp al, [b]  jb bIsGreater ; b > a  cmp al, [c] ; either a or c is max  jb cIsGreatest ; c > a  mov dl, al  jmp returnFindMax  bIsGreater: mov al, [c]  cmp al, [b]  jb bIsGreatest ; b < c  mov dl, [c] ; c is greatest  jmp returnFindMax  bIsGreatest: mov dl, [b]  jmp returnFindMax  cIsGreatest: mov dl, [c] ;dl = c  returnFindMax: mov [max], dl    ret  ;-----------------------------------------------------  ; FindTotal finds the sum of a,b and c  ;-----------------------------------------------------  FindTotal: mov al, 0  add al, [a]  add al, [b]  add al, [c]  mov [total], al  ret  ;-----------------------------------------------------  ; FindAvg calculates the Average of a,b and c  ; See 8088 Manual for the details of DIV Instruction  ;-----------------------------------------------------  FindAvg: mov al, [total]  mov cl, 3  div cl ;al = al/cl  mov [avg], al  ret  ;-----------------------------------------------------  main: call FindMin  call FindMax  call FindTotal  call FindAvg  ret  ;----------------------------------------------------  start: call main    MOV AX, 0x4C00 ; Terminate Program  INT 0x21 |

**Following is the code that we covered in class. It works on global variables, uses three numbers a, b and c, finds their min, max, total and avg and updates answer according to following conditions.**

; someFunc provides following functionality

; if ( avg > 0xC)

; ans = max

; else

; ans = min

; where avg, min, max and ans are global variables

|  |
| --- |
| ; SomeFunction with Global Variables  [ORG 0x0100]  jmp start  a: db 0xA  b: db 0xB  c: db 0xC  min: db 0xF  max: db 0x0  total: db 0x0  avg: db 0x0  ans: db 0x0  ;-----------------------------------------------------------  ; FindMin finds Min of global variables a, b and c  ; and saves it in global variable min  ;-----------------------------------------------------------  FindMin: mov al, [a]  cmp al, [b]  ja bIsSmaller ; b < a  cmp al, [c] ; either a or c is min  ja cIsSmallest ; a>c  mov dl, al  jmp return  bIsSmaller: mov al, [c]  cmp al, [b]  ja bIsSmallest ; b < c  mov dl, [c] ; c is smallest  jmp return  bIsSmallest: mov dl, [b]  jmp return  cIsSmallest: mov dl, [c] ;dl = c  return: mov [min], dl ;\*\*\* Which instruction will be executed next?  ;ret  ;-----------------------------------------------------------  ; FindMax finds Max of global variables a, b and c  ; and saves it in global variable max  ;-----------------------------------------------------------  FindMax: mov al, [a]  cmp al, [b]  jb bIsGreater ; b > a  cmp al, [c] ; either a or c is max  jb cIsGreatest ; c > a  mov dl, al  jmp returnFindMax  bIsGreater: mov al, [c]  cmp al, [b]  jb bIsGreatest ; b < c  mov dl, [c] ; c is greatest  jmp returnFindMax  bIsGreatest: mov dl, [b]  jmp returnFindMax  cIsGreatest: mov dl, [c] ;dl = c  returnFindMax: mov [max], dl ;\*\*\* Which instruction will be executed next?  ;ret  ;-----------------------------------------------------------  ; FindTotal finds Total of global variables a, b and c  ; and saves it in global variable total  ;-----------------------------------------------------------  FindTotal: mov al, 0  add al, [a]  add al, [b]  add al, [c]  mov [total], al ;\*\*\* Which instruction will be executed next?  ;ret  ;-----------------------------------------------------------  ; FindAvg finds Average of global variables a, b and c  ; by dividing total by 3  ; and saves it in global variable avg  ;-----------------------------------------------------------  FindAvg: mov al, [total]  mov cl, 3  div cl ;al = al/cl  mov [avg], al  jmp FindAns ;\*\*\* Which instruction will be executed next?  ;ret  ;-----------------------------------------------------------  ; someFunc provides following functionality  ; if ( avg > 0xC)  ; ans = max  ; else  ; ans = min  ; where avg, min, max and ans are global variables  ;-----------------------------------------------------------  someFunc: jmp FindMin  FindAns: cmp byte[avg], 0xC  jae AvgIsGreater  mov al, [min]  jmp ReturnFunc  AvgIsGreater: mov al, [max]  ReturnFunc: mov [ans], al ;\*\*\* Which instruction will be executed next?  ret ;\*\*\* Problem with commenting ret?  ;-----------------------------------------------------------  start: call someFunc  MOV AX, 0x4C00 ; Terminate Program  INT 0x21 |

Note: Properly Save and Restore State with each function call.

1. Update the function FindTotal such that it takes three parameters and returns their Sum. You are not allowed to use any global variable. Call FindTotal from main (start) and verify your result. Properly clean all the space used on Stack. (You only need to change a, b, c and total locations used in the function given above). After coming back from FindTotal save the returned value in DX.
2. Test FindTotal function on following data (one function call per test case). Try Step Over (F2) and Step Into (F1) commands for testing.
3. 0xA, 0xB, 0xC
4. 0x9, 0x1, 0x5
5. 0x2, 0xF, 0x8

|  |
| --- |
| Start: ; You have to run three test cases like this:  ; function call Test i  push 0xA  Push 0xB  Push 0xC  Call FindTotal  Pop DX  ; function call Test ii  push 0x9  Push 0x1  Push 0x5  Call FindTotal  Pop DX  ; function call Test iii  push 0x2  Push 0xF  Push 0x8  Call FindTotal  Pop DX  ;Termination Code here |

1. Update FindAvg such that it takes three parameters and returns two values, total and average. FindAvg will first call FindTotal (written in Activity 1) that takes three parameters and returns total. Then FindAvg will save the returned total in AX and divide it by a local variable that will be initialized to 3. Save the return values; total and average and return to main.
2. Call FindAvg function from Main on following data (one function call per test case) and verify its output. (AX = Total, BX=Average after returning from FindAvg)
3. 0xA, 0xB, 0xC
4. 0x9, 0x1, 0x5
5. 0x2, 0xF, 0x8