National University of Computer and Emerging Sciences



Laboratory Manual

for

Computer Organization and Assembly Language

|  |  |
| --- | --- |
|  |  |
|  |  |

**Department of Computer Science**

# COAL Lab 6 Manual

|  |
| --- |
| Objectives:  * Runtime stack * Modular approach, Defining & Using Procedures * Problems & Assignments |

# 6.1 Introduction

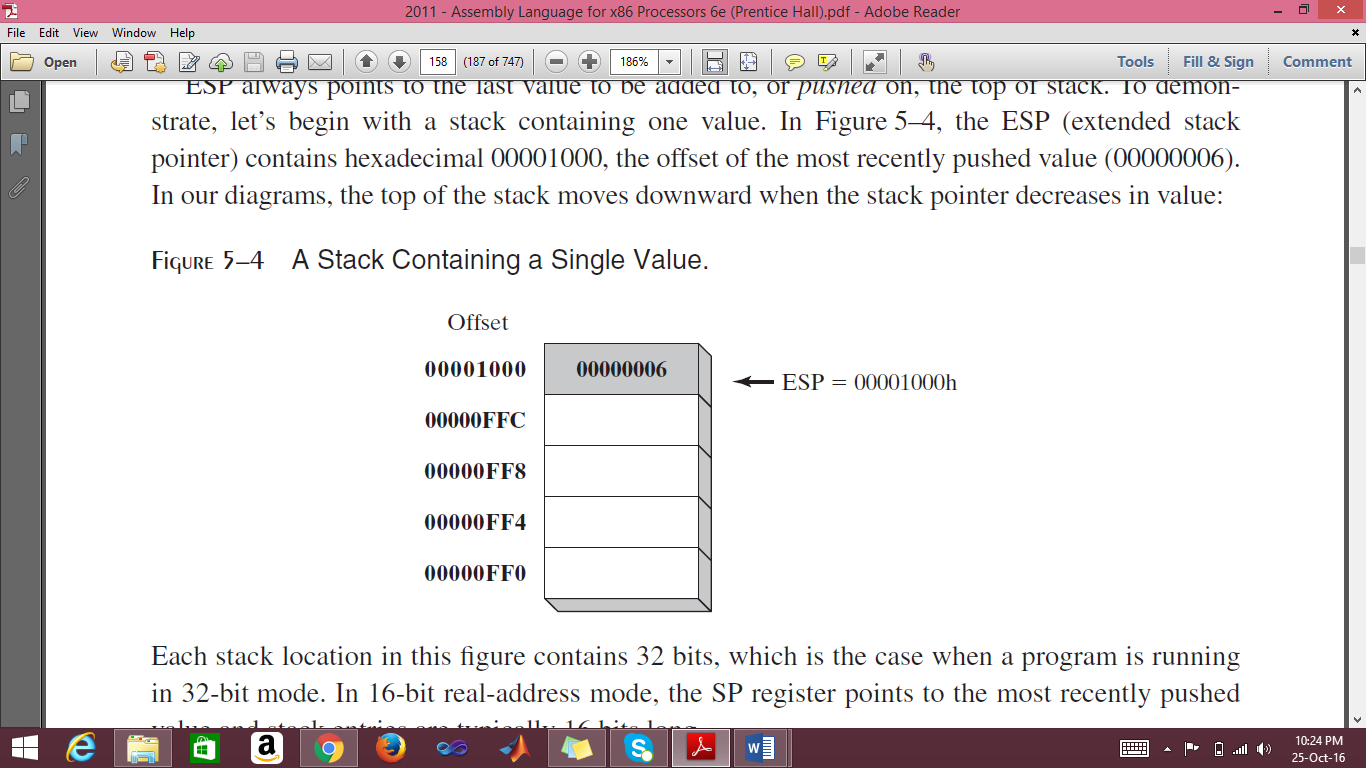
1. How to divide programs into manageable units by calling subroutines;
2. How programming languages use the runtime stack to track subroutine calls.

# 6.2 Irvine32 library

# 6.2 Runtime Stack

We concentrate specifically on the runtime stack. It is supported directly by hardware in the CPU, and it is an essential part of the mechanism for calling and returning from procedures. Most of the time, we just call it the stack.

The runtime stack is a memory array managed directly by the CPU, using the ESP register, known as the stack pointer register. The ESP register holds a 32-bit offset into some location on the stack. We rarely manipulate ESP directly; instead, it is indirectly modified by instructions such as CALL, RET, PUSH, and POP. ESP always points to the last value to be added to, or pushed on, the top of stack.



**Applications**

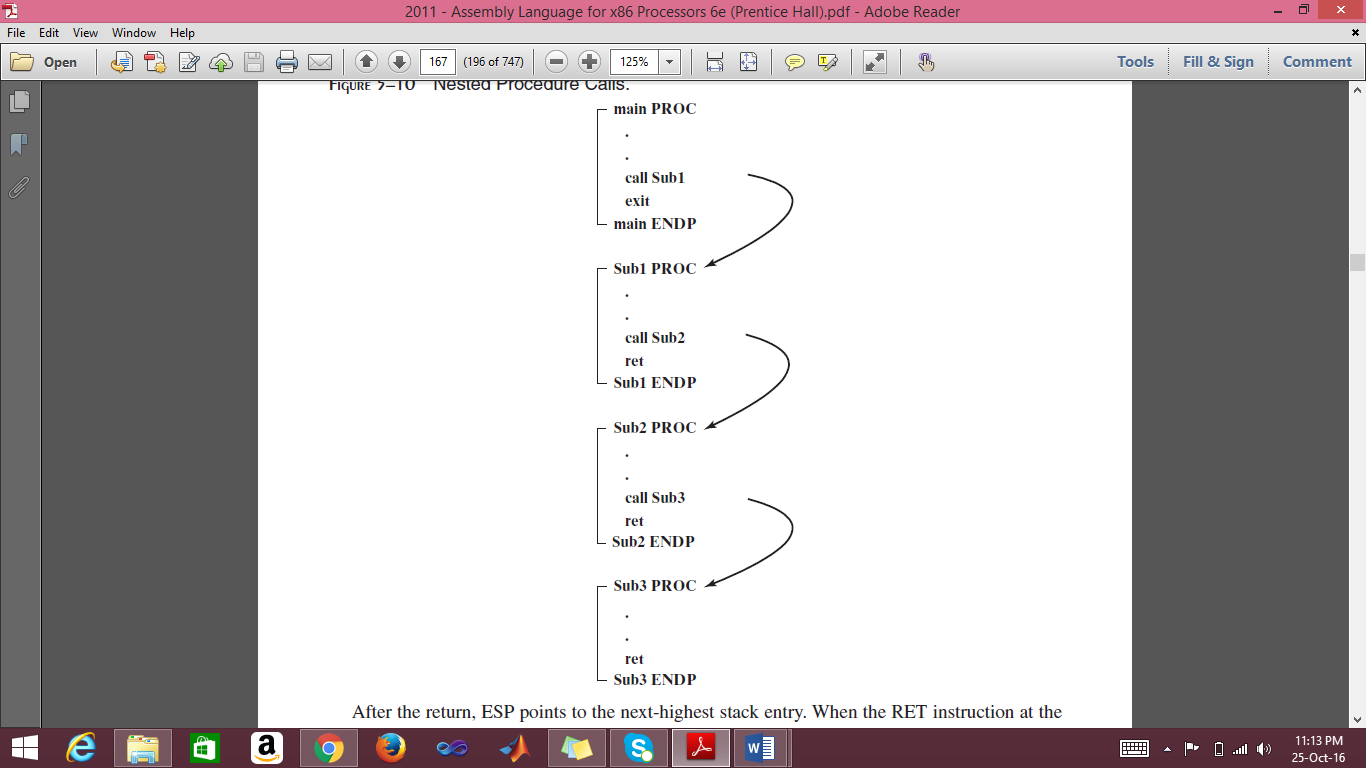
There are several important uses of runtime stacks in programs:

1. A stack makes a convenient temporary save area for registers when they are used for more than one purpose. After they are modified, they can be restored to their original values.
2. When the CALL instruction executes, the CPU saves the current subroutine’s return address on the stack.
3. When calling a subroutine, you pass input values called arguments by pushing them on the stack.
4. The stack provides temporary storage for local variables inside subroutines.

**Instructions**

|  |  |  |
| --- | --- | --- |
| Command | Syntax | Function |
| **PUSH** | PUSH *reg/mem32*  PUSH *imm32* | A 32-bit operand causes ESP to be decremented by 4. |
| **POP** | POP *reg/mem16*  POP *reg/mem32* | Copies the contents of the stack element pointed to by ESP into a 16- or 32-bit destination operand and then increments ESP by 2 (word size) by 4 (dword size). |
| **PUSHFD** | pushfd | Pushes the 32-bit EFLAGS register on the stack. |
| **POPFD** | popfd | Pops the stack into EFLAGS |
| **PUSHAD** | pushad | Pushes all of the 32-bit general-purpose registers on the stack in the following order: EAX, ECX, EDX, EBX, ESP (value before executing PUSHAD), EBP, ESI, and EDI. |
| **POPAD** | popad | Pops the same registers off the stack in reverse order |

# 6.4 Defining and Using Procedures



### USES Operator

The USES operator, coupled with the PROC directive, lets you list the names of all registers modified within a procedure. USES tells the assembler to do two things: First, generate PUSH instructions that save the registers on the stack at the beginning of the procedure. Second, generate POP instructions that restore the register values at the end of the procedure.

ArraySum PROC **USES esi ecx**

mov eax,0 ; set the sum to zero

L1:

add eax,[esi] ; add each integer to sum

add esi,TYPE DWORD ; point to next integer

loop L1 ; repeat for array size

ret ; sum is in EAX

ArraySum ENDP

## Problem(s) / Assignment(s)

|  |  |
| --- | --- |
| **Discussion & Practice** | **Estimated completion time: 1 hr, 30 mins** |

|  |  |
| --- | --- |
| **Problem 6.1: Write a program that take input string terminated by 0 (use procedure INPUT\_STR and Use ReadChar not WriteString), check whether the input string is palindrome or not. (Hint save the characters in stack also you got your problem solution).**  **Sample:**  **Use input string: MAAM**  **Yes the String is Palindrome** | **Estimated completion time:17 mins** |
|  |  |
|  |  |
| **Problem 6.2:** *Random Number Generator*  Write a program that should,   1. Randomly generate 10 unsigned integers in the range 0 to 4,294,967,294. 2. Next, it generates 10 signed integers in the range -50 to +49.   Use builtin procedures *Random32*, *Randomize*, *RandomRange* as shown below.  Follow modular approach i.e., make a procedure RAND\_U for unsigned integers and RAND\_S for signed integers. Your main should only be used for calling the procedures.    **Sample output:**  3221236194 2210931702 974700167 367494257 2227888607  926772240 506254858 1769123448 2288603673 736071794  -34 +27 +38 -34 +31 -13 -29 +44 -48 -43 | **Estimated completion time:20 mins** |

|  |  |  |  |
| --- | --- | --- | --- |
| Command | Input | Output | Function |
| **Random32** | Seed | EAX | Returns a 32-bit random integer. |
| **Randomize** |  |  | Initializes the starting seed value of the Random32 and RandomRange procedures. The seed equals the time of day, accurate to 1/100 of a second. |
| **RandomRange** | EAX= n | EAX | Produces a random integer within the range of 0 to n-1. |

***CODE:***

include irvine32.inc

.data

strr DB "Yes, it is a palindrome!!",0

strr2 DB "No, it is NOT a Palindrome!!",0

arr DB 100 dup(?),0

t1 DD ?

t2 DD ?

temp DD ?

.code

main proc

mov esi,OFFSET arr

mov ecx, Lengthof arr

call input\_string

l2:

mov al, '0'

cmp [esi], al

je equall

pop EAX

cmp [esi],al

jne notequal

inc esi

loop l2

equall:

call crlf

mov EDX, OFFSET strr

call writestring

jmp exitt

notequal:

call crlf

mov EDX, OFFSET strr2

call writestring

jmp exitt

exitt:

exit

main endp

input\_string PROC

mov t1, esi

mov t2, ecx

pop temp

l1:

call ReadChar

call writechar

mov [esi],Al

INC esi

cmp Al,'0'

je return

push EAX

loop l1

mov ecx, lengthof arr

return:

push temp

mov ecx, t2

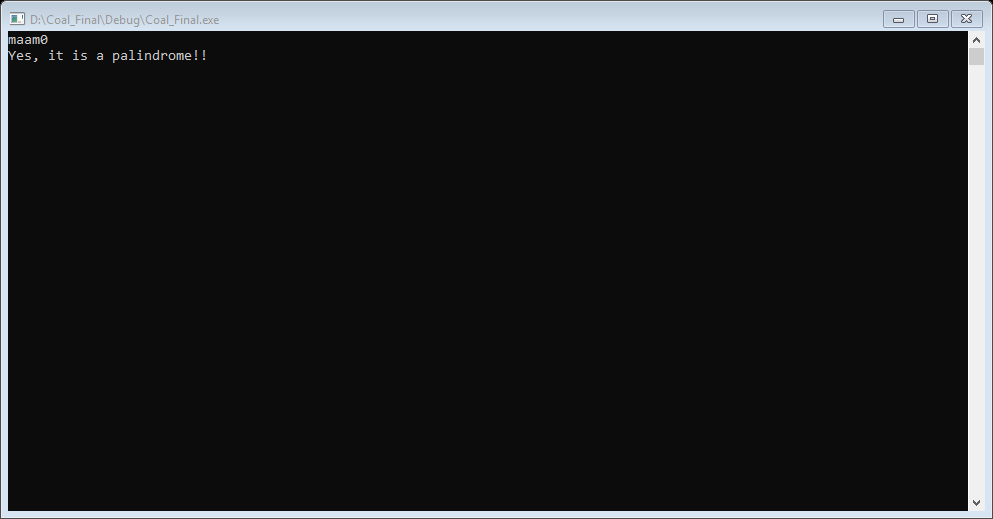
mov esi, t1

ret

input\_string endp

end main

**OUTPUT:**

****

**CODE:**

include Irvine32.inc

.code

main proc

call rand\_u

call rand\_s

exit

main endp

rand\_u proc

mov ecx, 10

l1:

mov eax, 4294967294

call randomrange

call writedec

call CRLF

loop l1

ret

rand\_u endp

rand\_s proc

mov ecx,10

L1:

mov eax,99

call randomrange

sub eax,50

call writeint

call crlf

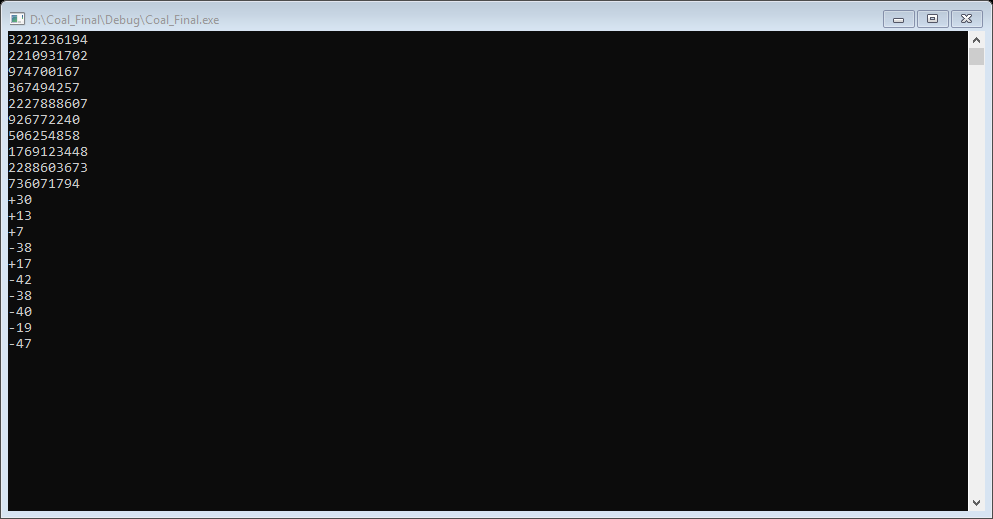
Loop L1

ret

rand\_s endp

end main

**OUTPUT:**

****