Assignment - 9

Title : Write ALP program to find factorial of given number.

Problem :
Write X86 ALP to find the factorial of a given integer number on a command line by using recursion. Explicit stack manipulation is expected in the code.

* Objective :- To understand how to use stack segment for recursion.

Outcome :- Students will study recursion using stack in ALP.

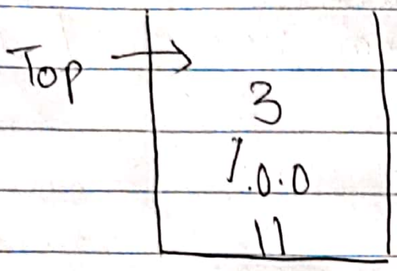
* S/W and H/W packages :
  - Processor : Core 2 duo / i3 / i5 / i7
  - OS : Linux 32 bit / 64 bit OS
  - Editor : gedit / vim
  - Assembler : NASM
  - Debugger : GDB

* Concept Related Theory :-
  PUSH - Push Operand onto the stack.

PUSH decrements the stack pointer by 2 if the operand-size attribute of the instructio is 16 bit, otherwise, it decrements the stack pointer by 4. PUSH then places the operand on the new top of stack, which is pointed to by the stack pointer.
The 80386 PUSH ESP instruction pushes the value of ESP as it existed before the instruction. This differs from the 8086, where PUSH SP pushes the new value (decremented by 2).

Instruction	Description
PUSH m16	Push memory word
PUSH m32	Push memory dword
PUSH r16	Push register word
PUSH r32	Push register dword
PUSH imm8	Push immediate byte
PUSH imm16	Push immediate word
PUSH imm32	Push immediate dword
PUSH CS	Push CS

* Pop – Pop a Word from the stack.
PoP replaces the previous contents of the memory, the register, or the segment register operand with the word on the top of the 80386 stack, addressed by SS:SP (address-size attribute of 16 bits) or SS:ESP (address size attribute of 32-bit). The stack pointer SP is incremented by 2 for an operand-size of 16-bits or by 4 for an operand-size of 32 bits. It then points to the new top of stack.



```
pop rbx
pop rbx
pop rbx
mov rsi,[rbx]
```

Instruction		Description.
POP	m 16	Pop top of stack into memory word.
POP	m 32	Pop top of stack into memory dword
POP	r 16	Pop top of stack into word register
POP	r 32	Pop top of stack into dword register
POP	DS	Pop top of stack into DS
POP	ES	Pop top of stack into ES
POP	SS	Pop top of stack into SS
POP	FS	Pop top of stack into FS
POP	GS	Pop top of stack into GS
~~POP~~		
~~POP~~		
~~POP~~		

* Algorithm
(1) Start
(2) Accept the number from user.
(3) Convert that number into Hexadecimal (ASCII to HEX)
(4) Compare accepted number with 1, if it is equal to 1 go to step 5, else push the number on stack and decrement the number and goto Step 4.
(5) Pop the content of stack and multiply with number.

(6) Repeat the step until stack becomes empty.
(7) Convert the number from HEX to ASCII
(8) Print the number.
(9) End.

Test Cases:

	Test Cases	Expected	Outcome	Result
1.	./a.out 05	78 H	As expected	Pass.
2.	./a.out 06	2D0 H	As expected	Pass
3.	./a.out 00	Factorial is 1	As expected	Pass
4)	./a.out ~~4~~09	58980 H	As expected	Pass.

Conclusion: Using the concept of recursion we successfully wrote an ALP program to find factorial of given number.