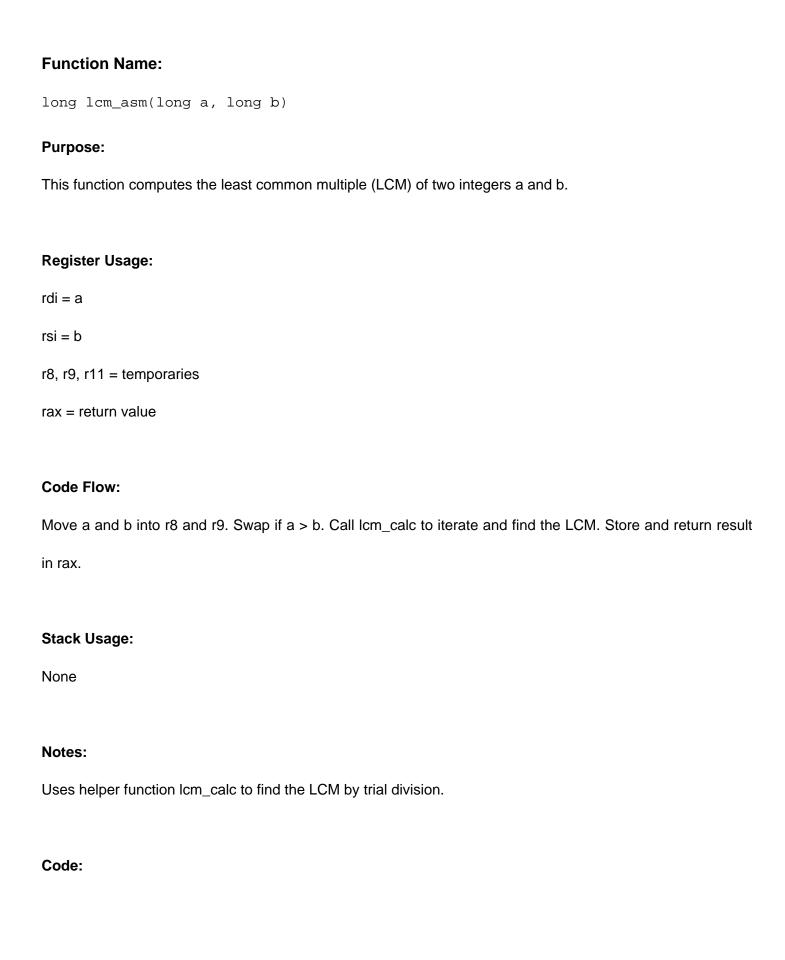


Submitted By:

Benchemam Ziad

Benzidoune Mohamed

Boucena Anes



```
lcm_asm:
          r8, rdi
   mov
          r9, rsi
   mov
          r8, r9
   cmp
   jle
          .call
           r11, r8
   mov
           r8, r9
   mov
           r9, r11
   mov
.call:
           r11, r9
           lcm_calc
   call
           r9, r11
   sub
           rax, r9
   mov
   ret
lcm_calc:
   xor
         rdx, rdx
          rax, r9
   mov
   div
          r8
   add
         r9, r11
           rdx, 0
   cmp
           lcm_calc
   jne
   ret
```

Function Name:

```
void revnum_asm(char* buf, int len)
```

Purpose:

Reverses a buffer of characters of length len.

Register Usage:

```
rdi = buf
rsi = len
r8 = start pointer
```

r10 = end pointer

r9, r11 = temporaries

Code Flow:

Calculate start and end pointers. Swap characters at both ends moving toward center. Loop until pointers meet.

Stack Usage:

rbp is pushed and popped for stack frame

Notes:

In-place reversal using two-pointer technique.

Code:

```
revnum_asm:
   push rbp
   mov rbp, rsp
   mov r8, rdi
        rcx, rsi
   mov
   lea
        r10, [r8 + rcx]
         r10
   dec
.rev_loop:
   cmp r8, r10
         .done
   jge
   movzx r9, byte [r8]
   movzx r11, byte [r10]
         byte [r10], r9b
   mov
```

```
mov byte [r8], r11b
inc r8
dec r10
jmp .rev_loop

.done:
   pop rbp
   ret
```

Function Name:

```
void rev_asm(char* str, int len)
```

Purpose:

Reverses a character array of length len in-place.

Register Usage:

```
rdi = str

rsi = len

r15 = pointer

rcx, rdx = indices
al, bl = temporary chars
```

Code Flow:

Two pointers are used from start and end of string. Characters are swapped until they meet.

Stack Usage:

Standard prologue and epilogue using rbp.

Notes:

Efficient two-pointer reverse algorithm.

Code:

```
rev_asm:
         rbp
   push
          rbp, rsp
   mov
          r15, rdi
   mov
          rcx, rsi
   mov
         rcx, rcx
   test
    jle
          .done
          rdx, rdx
   xor
   dec
          rcx
.loop:
          rdx, rcx
    cmp
          .done
    jge
          al, [r15 + rdx]
   mov
          bl, [r15 + rcx]
   mov
          [r15 + rdx], bl
   mov
          [r15 + rcx], al
   mov
    inc
          rdx
    dec
          rcx
    jmp
          .loop
.done:
          rbp
   pop
   ret
```