



Assembly

Submitted By:

Benchemam Ziad

Benzidoune Mohamed

Boucena Anes



Assembly Function Reports

Function Name:

`long lcm_asm(long a, long b)`

Purpose:

This function computes the least common multiple (LCM) of two integers a and b.

Register Usage:

`rdi` = a

`rsi` = b

`r8`, `r9`, `r11` = temporaries

`rax` = return value

Code Flow:

Move a and b into `r8` and `r9`. Swap if `a > b`. Call `lcm_calc` to iterate and find the LCM. Store and return result in `rax`.

Stack Usage:

None

Notes:

Uses helper function `lcm_calc` to find the LCM by trial division.

Code:

Assembly Function Reports

```
lcm_asm:
    mov     r8, rdi
    mov     r9, rsi
    cmp     r8, r9
    jle     .call
    mov     r11, r8
    mov     r8, r9
    mov     r9, r11
```

```
.call:
    mov     r11, r9
    call    lcm_calc
    sub     r9, r11
    mov     rax, r9
    ret
```

```
lcm_calc:
    xor     rdx, rdx
    mov     rax, r9
    div     r8
    add     r9, r11
    cmp     rdx, 0
    jne     lcm_calc
    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

Assembly Function Reports

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
    mov     rcx, rsi
    lea     r10, [r8 + rcx]
    dec     r10

.rev_loop:
    cmp     r8, r10
    jge     .done
    movzx   r9, byte [r8]
    movzx   r11, byte [r10]
    mov     byte [r10], r9b
```

Assembly Function Reports

```
    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:

Assembly Function Reports

Efficient two-pointer reverse algorithm.

Code:

```
rev_asm:
    push    rbp
    mov     rbp, rsp
    mov     r15, rdi
    mov     rcx, rsi
    test    rcx, rcx
    jle     .done
    xor     rdx, rdx
    dec     rcx

.loop:
    cmp     rdx, rcx
    jge     .done
    mov     al, [r15 + rdx]
    mov     bl, [r15 + rcx]
    mov     [r15 + rdx], bl
    mov     [r15 + rcx], al
    inc     rdx
    dec     rcx
    jmp     .loop

.done:
    pop     rbp
    ret
```


