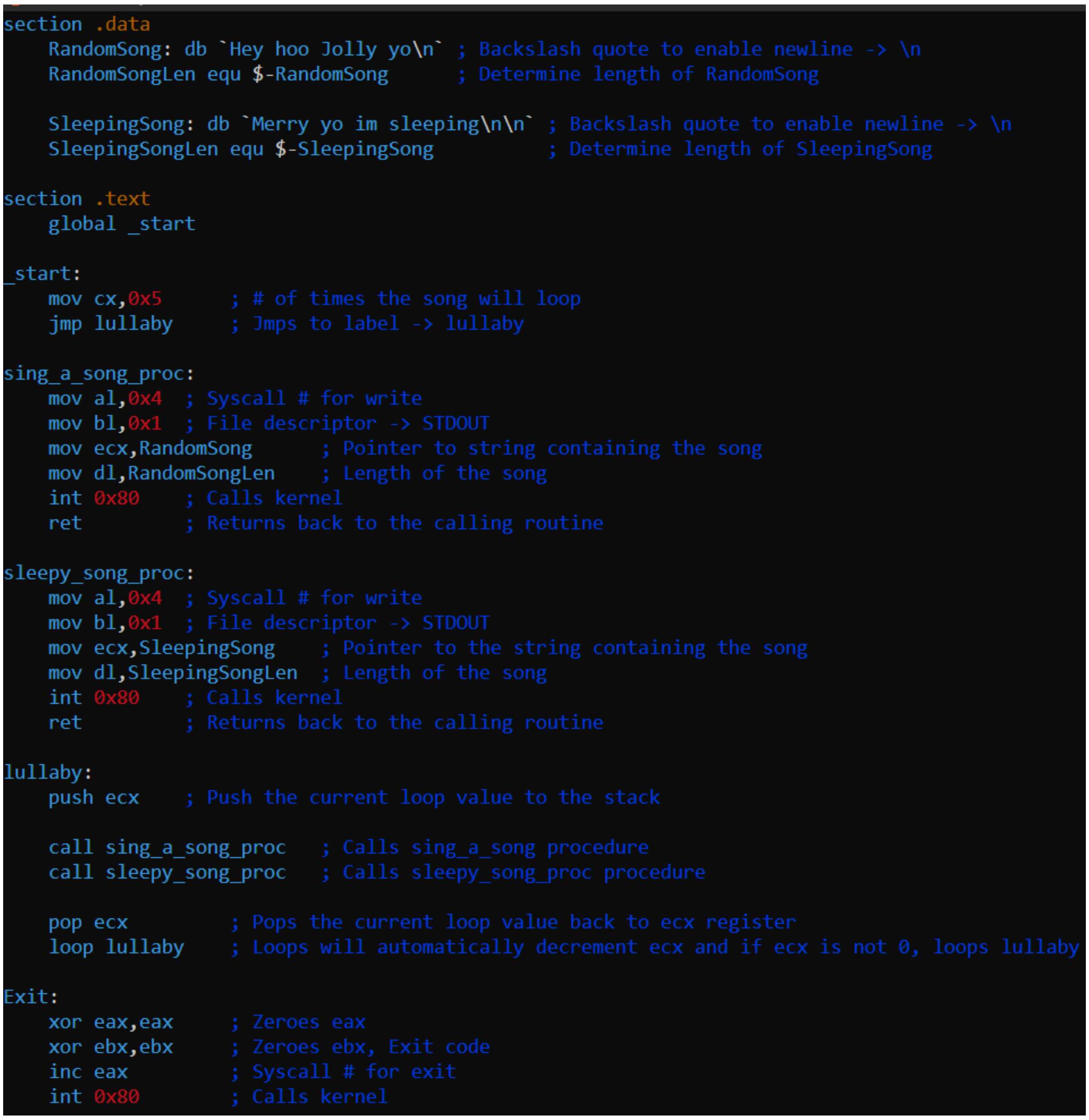
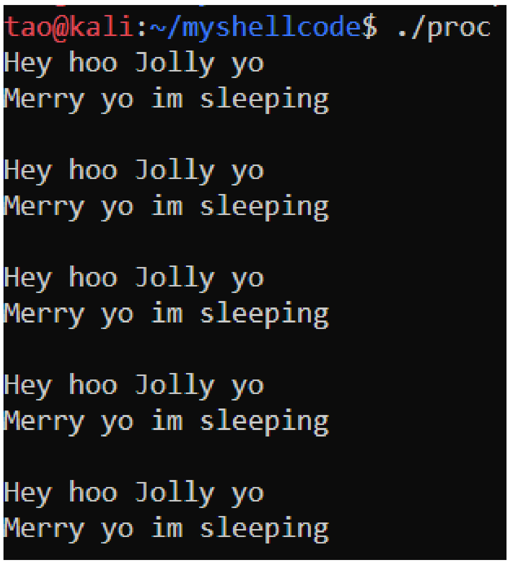
This program calls 2 different procedure with 2 different messages(songs)



```
section .data
    RandomSong: db `Hey hoo Jolly yo\n` ; Backslash quote to enable newline -> \n
    RandomSongLen equ $-RandomSong      ; Determine length of RandomSong

    SleepingSong: db `Merry yo im sleeping\n\n` ; Backslash quote to enable newline -> \n
    SleepingSongLen equ $-SleepingSong          ; Determine length of SleepingSong

section .text
    global _start

_start:
    mov cx,0x5      ; # of times the song will loop
    jmp lullaby     ; Jmps to label -> lullaby

sing_a_song_proc:
    mov al,0x4  ; Syscall # for write
    mov bl,0x1  ; File descriptor -> STDOUT
    mov ecx,RandomSong      ; Pointer to string containing the song
    mov dl,RandomSongLen    ; Length of the song
    int 0x80    ; Calls kernel
    ret         ; Returns back to the calling routine

sleepy_song_proc:
    mov al,0x4  ; Syscall # for write
    mov bl,0x1  ; File descriptor -> STDOUT
    mov ecx,SleepingSong    ; Pointer to the string containing the song
    mov dl,SleepingSongLen  ; Length of the song
    int 0x80    ; Calls kernel
    ret         ; Returns back to the calling routine

lullaby:
    push ecx    ; Push the current loop value to the stack

    call sing_a_song_proc   ; Calls sing_a_song procedure
    call sleepy_song_proc   ; Calls sleepy_song_proc procedure

    pop ecx         ; Pops the current loop value back to ecx register
    loop lullaby    ; Loops will automatically decrement ecx and if ecx is not 0, loops lullaby

Exit:
    xor eax,eax     ; Zeroes eax
    xor ebx,ebx     ; Zeroes ebx, Exit code
    inc eax         ; Syscall # for exit
    int 0x80        ; Calls kernel
```

Output



```
tao@kali:~/myshellcode$ ./proc
Hey hoo Jolly yo
Merry yo im sleeping

Hey hoo Jolly yo
Merry yo im sleeping

Hey hoo Jolly yo
Merry yo im sleeping

Hey hoo Jolly yo
Merry yo im sleeping

Hey hoo Jolly yo
Merry yo im sleeping
```