```
section .bss
    temp_input resb 4  ; Buffer to store temperature input
    ac_control resb 2  ; Buffer to strore AC control input(on/off)

section .data
    prompt_control db "Turn AC On or Off? (Enter '1' for On, '0' for Off): ", 10, 0
    prompt_control_len equ $ - prompt_control

    prompt db "Enter current temperature(°C): ", 0
    prompt_len equ $ - prompt
    threshold equ 18  ; Threshold temperature to turn on AC

    msg_ac_off db "AC is turned OFF.", 10, 0
    msg_ac_off_len equ $ - msg_ac_off
    msg_turnon db "AC turning on...Room is hot!", 10, 0
    msg_standby db "AC standing by,keeping ventilator on...Temperature equal to 18°C", 10,
0
    msg_turnoff db "AC turning off...Room too cold!", 10, 0

section .text
global _start

_start:

    ; Prompt user to turn AC on or Off
    mov rax, 1                  ; syscall: write
    mov rdi, 1                  ; file descriptor: stdout
    mov rsi, prompt_control     ; buffer: prompt_control
    mov rdx, prompt_control_len ; buffer length: prompt_control_len
    syscall

    ; Read AC control input
    mov rax, 0                  ; syscall: read
    mov rdi, 0                  ; file descriptor: stdin
    mov rsi, ac_control         ; buffer: ac_control
    mov rdx, 2                  ; buffer length: 2 (Including newline)
    syscall

    ; Check AC control input
    movzx rcx, byte [ac_control]        ; Load AC control input
    cmp rcx, '1'                        ; Compare with '1' (AC On)
    je .ac_on

    cmp rcx, '0'
    je .ac_off

    ; If invalid input, repeat the whole block
    jmp _start

.ac_off:
    ; Print AC Off message
    mov rax, 1                  ; syscall: write
    mov rdi, 1                  ; file descriptor: stdout
    mov rsi, msg_ac_off         ; buffer: msg_ac_off
    mov rdx, msg_ac_off_len     ; buffer length
    syscall

    jmp end_program

.ac_on:
    ; Print the prompt
    mov rax, 1                  ; syscall: write
    mov rdi, 1                  ; file descriptor: stdout
    mov rsi, prompt             ; buffer: prompt
    mov rdx, prompt_len         ; buffer length: prompt_len
    syscall

    ; Read user input(Temperature)
    mov rax, 0                  ; syscall: read
    mov rdi, 0                  ; file descriptor: stdin
    mov rsi, temp_input         ; buffer: temp_input
    mov rdx, 4                  ; buffer length: 4 (including newline)
    syscall

    ; Convert ASCII input to integer
    xor rax, rax                ; Clear rax
    xor rcx, rcx                ; Clear rcx for loop counter
    mov rbx, temp_input         ; Point rbx to temp_input buffer

convert_loop:
    movzx rdx, byte [rbx+rcx]   ; Load byte from input buffer
    test rdx, rdx               ; Check if byte is null terminator or newline
    jz compare_temperature      ; If null terminator, end loop
    cmp rdx, 10                 ; Check if newline
    je compare_temperature
    sub rdx, '0'                ; Convert ASCII to integer
    imul rax, rax, 10           ; Multiply rax by 10
    add rax, rdx                ; Add the digit to rax
    inc rcx                     ; Move to the next character
    jmp convert_loop            ; Repeat the loop

compare_temperature:
    ; Compare the temperature to 18°C
    cmp rax, threshold
    jl .less
    je .equal

.greater:
    ; Print "Greater than Threshold message"
    mov rax, 1                  ; syscall: write
    mov rdi, 1                  ; file descriptor: stdout
    mov rsi, msg_turnon         ; buffer: msg_turnon
    mov rdx, 30                 ; buffer length: 30 including newline
    syscall
    jmp end_program

.less:
    ; Print "Less than threshold message"
    mov rax, 1                  ; syscall: write
    mov rdi, 1                  ; file descriptor: stdout
    mov rsi, msg_turnoff        ; buffer: msg_turnoff
    mov rdx, 33                 ; buffer length: 33 including newline
    syscall
    jmp end_program

.equal:
    ; Print "Equal to threshold message"
    mov rax, 1                  ; syscall: write
    mov rdi, 1                  ; file descriptor: stdout
    mov rsi, msg_standby        ; buffer: msg_standby
    mov rdx, 66                 ; buffer length: 66 including newline
    syscall
    jmp end_program



end_program:
    ; Exit the program
    mov rax, 60                 ; syscall: exit
    xor rdi, rdi                ; status: 0
    syscall
```