## TASK 1:

org 0x100 ; Set the origin to 0x100

jmp start ; Jump to the start label

multiplicand: db 13 ; Define the multiplicand (13 in decimal)

multiplier: db 5 ; Define the multiplier (5 in decimal)

result: db 0 ; Initialize the result to 0

start:

mov cl, 4 ; Set the loop counter to 4

mov bl, [multiplicand] ; Load the multiplicand into BL

mov dl, [multiplier] ; Load the multiplier into DL

checkbit:

shr dl, 1 ; Shift the least significant bit of the multiplier into the carry flag

jnc skip ; If the carry flag is clear (no carry), skip the addition

add [result], bl ; Add the multiplicand to the result

skip:

shl bl, 1 ; Shift the multiplicand left by 1

dec cl ; Decrement the loop counter

jnz checkbit ; Repeat the loop until the counter is not zero

mov ax, 0x4c00 ;

int 0x21

|  |  |  |  |
| --- | --- | --- | --- |
| **Count** | **Multiplicand** | **Multiplier** | **Result** |
| 4 | 00001101 | 00000101 | 00000000 ; Initial values in binary |
| 3 | 00001101 | 00000010 | 00001101 ; Shifted right, added 00001101 (13 in binary) |
| 2 | 00011010 | 00000001 | 00011010 ; Shifted right, added 00011010 (26 in binary) |
| 1 | 00110100 | 00000000 | 00100111 ; Shifted right, added 00110100 (39 in binary) |
| 0 | 01101000 | 00000000 | 01000001 ; Shifted right, added 01101000 (52 in binary) |
| Final | 11010000 | 00000000 | 01000001 ; Shifted right, no addition (Final result: 65 in binary) |

## TASK2:

org 0x0100

jmp start

start:

mov cx, 16

mov dx, [multiplier]

checkbit:

shr dx, 1

jnc skip

mov ax, [multiplicand]

add [result], ax

mov ax, [multiplicand+2]

adc [result+2], ax

skip:

shl word [multiplicand], 1

rcl word [multiplicand+2], 1

dec cl

jnz checkbit

mov ax, 0x4c00

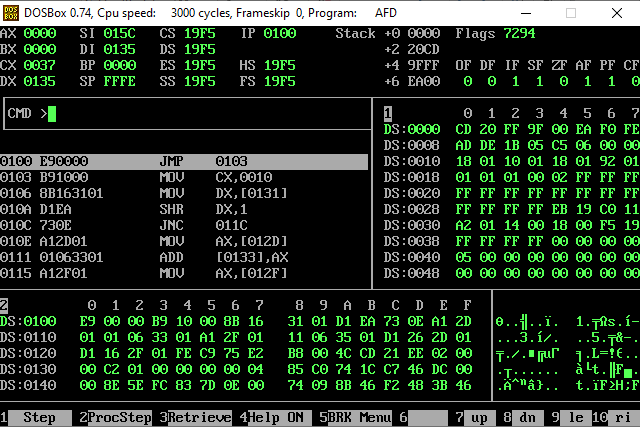
int 0x21

multiplicand: dd 750 ;

multiplier: dw 450 ;

result: dd 0 ;

# OUTPUT:



# TASK 3:

org 0x0100

jmp start

start:

mov si, num

mov di, count\_address

mov cx, 0

count\_loop:

mov ax, [si]

mov bx, ax

mov dx, di

count\_bits:

rol bx, 1

jnc no\_increment

inc di

no\_increment:

shl ax, 1

inc si

dec cx

jnz count\_bits

add di, 2

inc di

dec cx

jnz count\_loop

mov ax, 4C00h

int 21h

num dw 65535, 61347, 39321, 41279

count dw 0

count\_address dw 0

OUTPUT:  
