Q1. Assembly language program to find largest number in an Array.

```
section .data
    array db 10, 5, 8, 2, 15, 3
    array_size equ $-array

section .text
    global _start

_start:
    mov esi, array
    mov cl, [esi]
    inc esi
    dec array_size

loop_start:
    cmp byte [esi], cl
    jle skip_update
    mov cl, [esi]

skip_update:
    inc esi
    dec array_size
    jnz loop_start
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

Q2. Assembly language program to find smallest number in an array.

```
section .data
    array db 10, 5, 8, 2, 15, 3
    array_size equ $-array

section .text
    global _start

_start:
    mov esi, array
    mov cl, [esi]
    inc esi
    dec array_size

loop_start:
    cmp byte [esi], cl
    jl update_smallest
    inc esi
    dec array_size
```

```
    jnz loop_start
    jmp done

update_smallest:
    mov cl, [esi]
    inc esi
    dec array_size
    jnz loop_start

done:

    mov eax, 1
    xor ebx, ebx
    int 0x80
```

Q3. Assembly language program for adding to two arrays

```
section .data
    array1 db 1, 2, 3, 4, 5
    array2 db 6, 7, 8, 9, 10
    array_size equ 5

section .bss
    result_array resb 5

section .text
    global _start

_start:
    mov esi, array1
    mov edi, array2
    mov ebx, result_array
    mov ecx, array_size

add_arrays:
    mov al, [esi]
    add al, [edi]
    mov [ebx], al
    inc esi
    inc edi
    inc ebx
    loop add_arrays

    mov eax, 1
    xor ebx, ebx
    int 0x80
```

Q4.) Assembly language program to separate even and odd numbers from an array.

```
section .data
    array db 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
    array_size equ 10

section .bss
    even_array resb 10
    odd_array resb 10

section .text
    global _start

_start:
    mov esi, array
    mov edi, even_array
    mov ebx, odd_array
    mov ecx, array_size

separate_numbers:
    mov al, [esi]
    test al, 1
    jz store_even
    mov [ebx], al
    inc ebx
    jmp next_iteration

store_even:
    mov [edi], al
    inc edi

next_iteration:
    inc esi
    loop separate_numbers


    mov eax, 1
    xor ebx, ebx
    int 0x80
```

Q5. Assembly language rogram to find prime numbers between a given range

```
section .data
    start_number dw 10
    end_number dw 30
    newline db 10
```

```
section .text
    global _start

_start:
    mov cx, start_number

check_prime:
    mov bx, 2
    mov ax, cx

prime_loop:
    cmp bx, ax
    jg next_number

    mov dx, 0
    div bx

    cmp dx, 0
    je next_number

    inc bx
    jmp prime_loop

next_number:
    cmp ax, cx
    jne increment

    mov eax, 4
    mov ebx, 1
    mov ecx, cx
    mov edx, 2
    int 0x80

    mov eax, 4
    mov ebx, 1
    mov ecx, newline
    mov edx, 1
    int 0x80

increment:
    inc cx
    cmp cx, end_number
    jle check_prime

    mov eax, 1
    xor ebx, ebx
    int 0x80
```

Q6. Assembly language program to find factorial of the given number.

```
section .data
    number dw 5

section .text
    global _start

_start:
    mov cx, number
    mov ax, 1

factorial_loop:
    mul cx
    loop factorial_loop

    mov eax, 1
    xor ebx, ebx
    int 0x80
```