4. Declare 2 different variables and compare those if the numbers are equal or not.

```
.model small
.stack 100h
.data

        var db 5

        var1 db 5
.code

        MOV AX,@DATA
        MOV DS,AX

        mov bl,var
        mov bh,var1

        cmp bl,bh

        je exit

        mov bl,8

exit:

mov ah,4ch
int 21h
end
```

5. Store numbers 0 to 9 in array at different indices.

```
.model small
.stack 100h
.data

        Array db 10 dup (?)

.code
        mov ax, @DATA
        mov ds, ax

        mov si,0
        mov al,0

start:

        mov cx,10

        mov Array[si],al

        inc al
        inc si

loop Start
```

```
        mov si,offset Array

mov ah,4ch
int 21h
end
```

6. Store numbers a to z in array at different indices

```
.model small
.stack 100h
.data

        Array db 26 dup (?)

.code
        mov ax, @DATA
        mov ds, ax

        mov si,0
        mov al,61H


start:



        mov cx,26

        mov Array[si],al
```

```
        inc al
        inc si


loop Start

        mov si,offset Array

mov ah,4ch
int 21h
end
```

7. Store numbers A to Z in array at different indices

```
.model small
.stack 100h
.data

        Array db 26 dup (?)

.code
        mov ax, @DATA
        mov ds, ax

        mov si,0
        mov al,41H


start:
```

```
        mov cx,26

        mov Array[si],al

        inc al
        inc si


loop Start

        mov si,offset Array

mov ah,4ch
int 21h
end
```

8. Perform a check on a number if that is even or odd.

```
.model small
.stack 100h
.data

        var dw 12H

.code

        mov ax, @DATA
        mov ds, ax

        mov ax,var
```

```
        mov bl,2

        div bl
        mov dl,0

        cmp dl,ah
        je exit

        mov bl,5

exit:


mov ah,4ch
int 21h
end
```

9. Declare an array of 6 elements; perform a check if the elements are even or odd. If even save character 'e' in a variable and if odd save character 'o' in that variable

```
.model small
.stack 100h
.data

        Array dw 2,5,6,7,2,5

.code
        mov ax, @DATA
        mov ds, ax
```

```
        mov si,0
        mov cx,6

start:

        mov ax,Array[si]
        mov bl,2

        div bl

        cmp ah,0
        jn Even1

        cmp ah,0
        jne Odd1


Even1:

        mov Array[si],'e'
        inc al
        inc si
        jmp exit
Odd1:

        mov Array[si],'o'
        inc al
        inc si
        jmp exit
```

```
exit:
loop Start

        mov si,offset Array

mov ah,4ch
int 21h
end
```

10. Declare an array of 5 elements; find the sum of all elements in the array and save the sum in a variable using loop and indirect addressing.

```
.model small
.stack 100h
.data

        Array dw 1H,2H,3H,4H,5H
        var db ?

.code
        mov ax, @DATA
        mov ds, ax

        mov si,offset Array

        mov cx,5

start:
```

```
        add bl,[si]

        inc si


loop Start

        mov var,al

        mov si,offset var

mov ah,4ch
int 21h
end
```