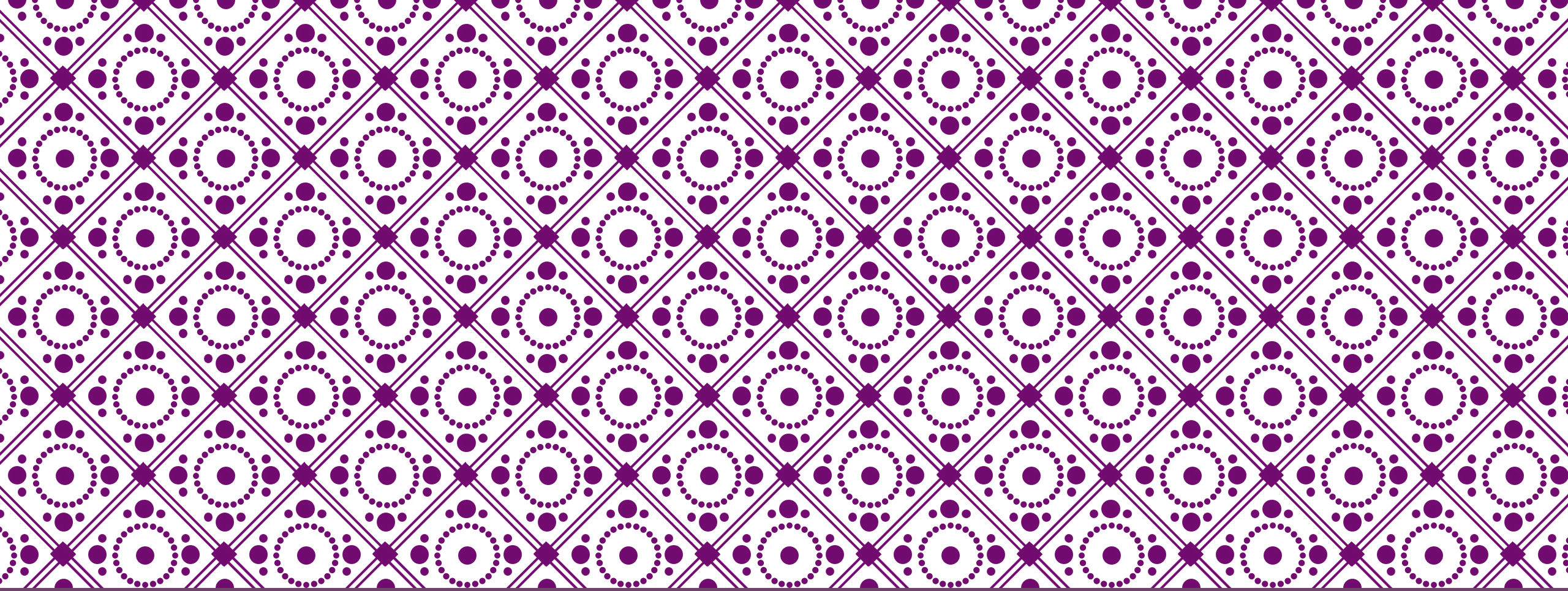


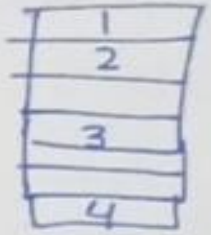
# ARRAYS AND STRING

Let suppose you have 3 numbers & want to store as variable, like

```
Var1 db 1
Var1 db 2
Var3 db 3
```

need 3 Variables for 3 numbers.

When we initialize, RAM have any empty space it store this number, means 3 numbers save randomly in RAM.
NO Sequence, No Order.



RAM.

Better to use one variable for all values, it remain sequence in RAM, & we easily access with one variable, we need array for that.

Why need Array :- "To store many characters with single variable name in sequence in memory."

"Array collection of characters in sequence".

**Where Array is Initialized :-** Array is Variable so it is initialize in .data directive.

"Array is defined in .data directive of program as Variable"

**How to Initialize** e.g. 1,2,3,4,5,6.

In same way as Variable but with multiple Values.

```
Arr1 db 1,2,3,4,5,6.
Arr1 db 'a','b','c' or 'abc' of characters.
Arr1 db 'abc'
Arr1 db 'a','a','a'
Arr1 db ?,?,?
```

for unassigned not given value only reserve memory, (Uninitialized)

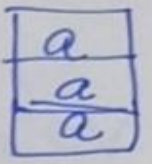
If same character comes for many times or so, In assembly facility of duplicate. (dup)

```
Arr1 db 3 Dup('a')
```

3 → no. of characters

Dup('a') → Duplicates the value.

If 3, 3 boxes in RAM




If Unassigned, Arr1 db 4 Dup(?)

For example `arr1 db 1,2,3,4`

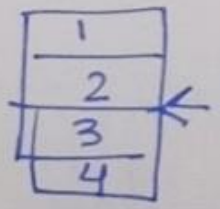
## How To Access Array.

How we Access Variable we write two statements

1) Send Data address.

1). Data segment get the address to initialize heap memory & quickly access variable.

```
Mov ax, @data
Mov ds, ax.
```

In RAM, 

so we send it to this address to Access Variable here.

We need Register to Access Address or Hold Address.

## Source Index Register (SI)

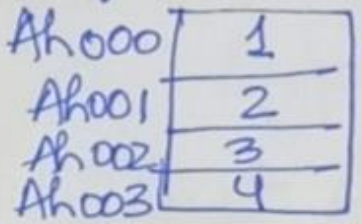
SI hold address of first variable like 1,

"Source Index Register used as pointer to Access Array"

We give it addres of 1st variable value, move the addres to SI, we use offset (to move addres).

Mov si, offset arr1

Offset 1 means starting address of first character.




∴ So We sent address to SI, SI now find address of first character.

Now we easily used & print it.

To print → mov dx, si X because SI has address.

We write → mov dx, [si] Bracket form to access value at address.

mov ah, 2

int 21h

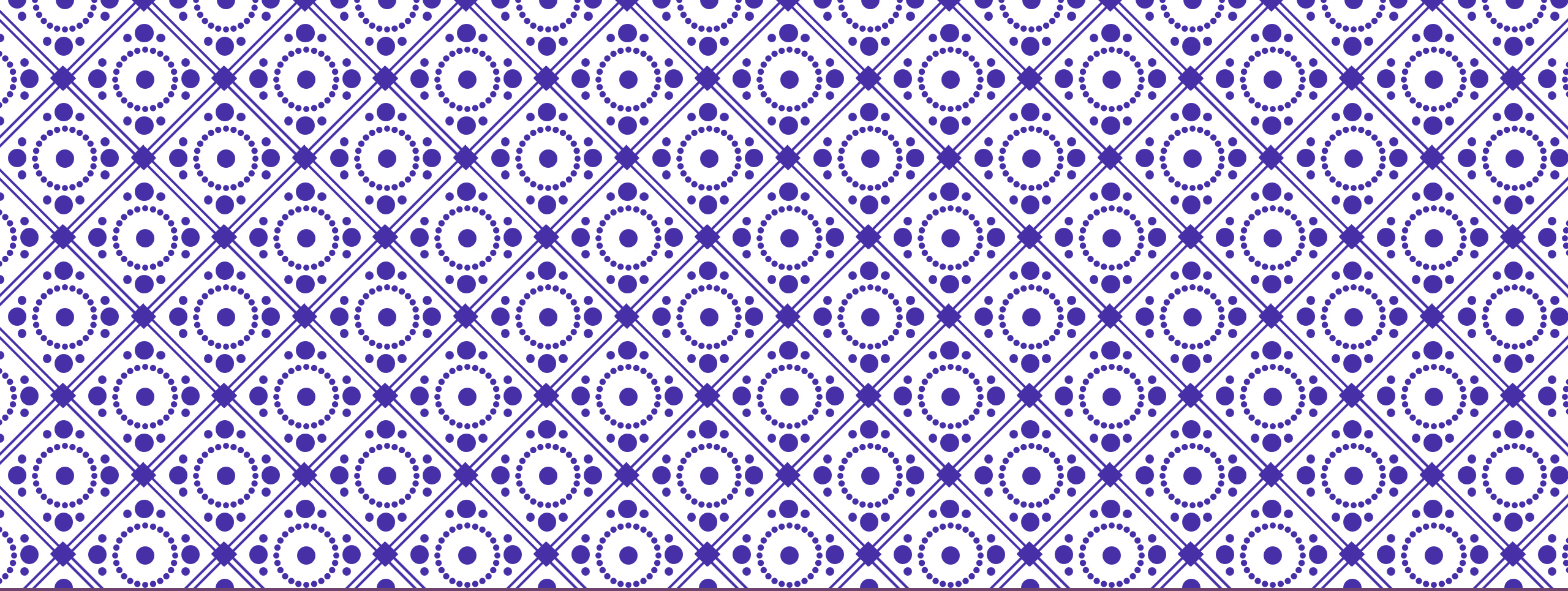
## To increment in Value.

```
mov dx, [si + 1]
```

→ Because it is in sequence.

or

```
inc si
```



# LAB

PROGRAM TO PRINT ARRAY USING LOOP

# Program to Print an Array Using Loop.

Why using loop to Access array?

→ Suppose, array db 'a', 'b', 'c'

Only 3 letters in this case, if many letters/characters, we always print and increment, so fast way is loop,

With the help of loop Access Array.

```
.data
array db 'a', 'b', c
.code
main proc
mov ax, @data        } To Access Variable
mov ds, ax           } in .data.
```

```
mov si, offset array
```
→ starting address of array go to si, it point to array.

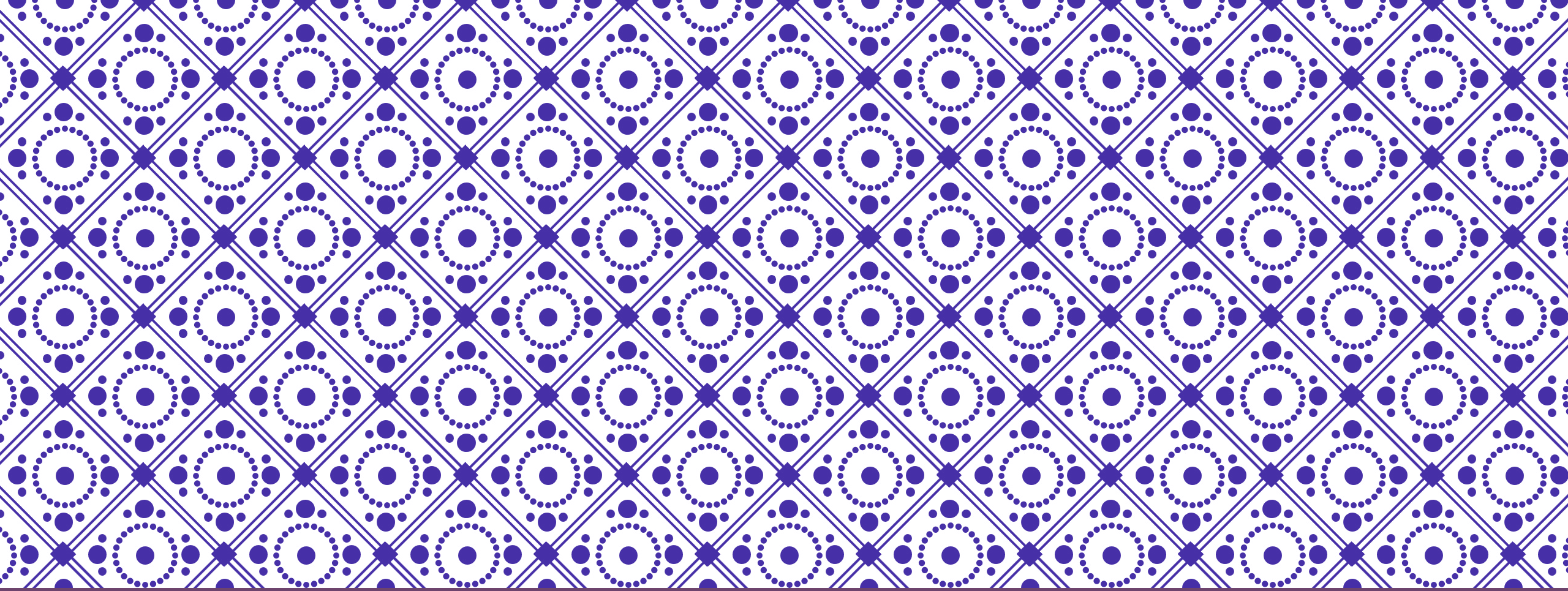
```
mov cx, 3
```
First we decide how many times to run loop. send to mov cx.

Now loop structure.

```
L1:
mov dx, [si]
```
→ ~~Address~~ mov address of si i.e. a, [ ] used for value at address.

```
mov ah, 2
int 21h
inc si
loop L1
```
→ To go to next value, increment in si

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



# LAB

PROGRAM TO INPUT STRING AND PRINT IT

# - Program to Input String and Print it

→ String is variable, combination of characters, Parsage or Paragraph. In end $ is must for string. (Because when we print with help of function 9, function 9 check till where it see $ sign, stop printing of characters.

## Input Any string?

Array is also type of string

- Practice of Jump, cmp, conditional jum, Unconditional jump to Input String

When we take input string we want $ is must, But user don't give $, just press enter key, so the input of user we have store it in array & give $ by ourselves, so when it is printed it print as string.

We used Array → 100 character limit for user Input.

```
Var1 db 100 dup('$')
```

Take Variable, give size, decide how many characters needed.
We need $ at end so used $ instead of unassigned?

If user give abc then
other 97 letters are $$$....., abc$$$$....
it only print abc...

---

In .data we initialize array.

```
.data
Var1 db 100 dup('$')

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

} quickly access variable. heap memory initialized

```
mov si, offset var1 ⟶ To access array starting add. of var1 now go to si
l1: ⟶ Start label,
mov ah, 1
int 21h ⟶ To take input from user.
cmp al, 13 ⟶ Now we send i/p to si, but user write we have to store all values until Enter key is press so we compare the input which is in al with ascii code of enter key which is 13
je programend ⟶ If equal je to program end label.
mov [si], al ⟶ If input not equal to 13, (no Enter key) mov input to si, inc si & again jump to label l1 for input.
inc si
jump l1
```



```
Programend: ⟶ the initialized array is printed. & program is end.
mov dx, offset var1
mov ah, 9
int 21h
mov ah, 4ch ⟶ Return to Dos.
int 21h
main endp
end main
```