INDEXED ADDRESSING

- USE REGISTERS SI OR DI TO ACCESS ELEMENTS IN AN ARRAY

E.G.
```
.DATA
MYARRAY   DW   10, 20, 30, 40, 50
```
TWO BYTES FOR EACH ARRAY ELEMENT

- CODE

INDEXED ADDRESSING

```
MOV   SI, 0
MOV   AX, MYARRAY[SI]
CALL  OUTDEC
```
PRINT OUT 10

HOW WOULD YOU PRINT OUT 20?
- NEED TO ADJUST SI — BY WHAT AMOUNT — BY 2.

E.G.
```
ADD   SI, 2
MOV   AX, MYARRAY[SI]
CALL  OUTDEC
```
PUT CODE BLOCK IN LOOP TO PRINT OUT ALL ARRAY ELEMENTS

BASED INDEXED ADDRESSING

- FOR EXAMPLE

- DATA

```
MYARRAY   (DW)   10,20,30,40,50
```

- CODE

```
MOV   SI,0
MOV   BX, OFFSET MYARRAY ; BASE
MOV   AX, [BX][SI]
CALL  OUTDEC
```

WHAT GETS PRINTED? 10

OFFSET OF ARRAY

BX: BASE

SI OR DI: INDEX INTO THE ARRAY

HOW DO WE PRINT 20?

```
ADD   SI, 2
MOV   AX, [BX][SI]
CALL  OUTDEC
```

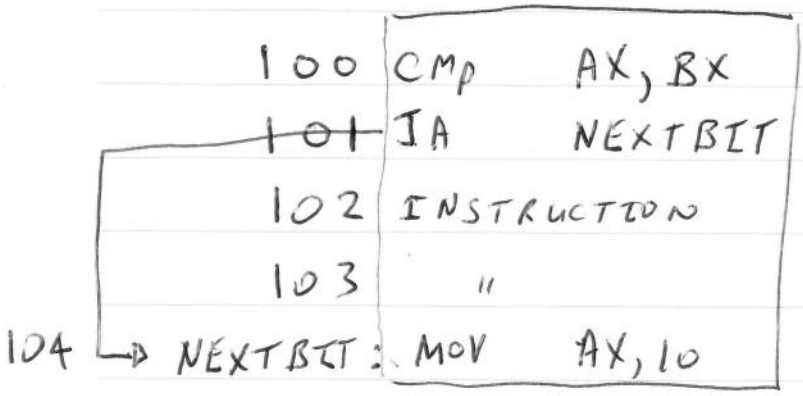
LOOP AROUND THIS CODE TO PRINT ALL ARRAY ELEMENTS.

PC RELATIVE ADDRESSING

PC : PROGRAM COUNTER
(REGISTER THAT STORES ADDRESS OF NEXT INSTRUCTION TO BE EXECUTED)

IF YOU HAVE A JUMP INSTRUCTION, THE JUMP IS RELATIVE TO THE PROGRAM COUNTER'S VALUE.

E.G.



JA IS ENCODED AS

JA +3

IF THE TARGET WE ARE JUMPING TO WAS BACKWARDS, INSTRUCTION IS ENCODED AS

FOR E.G.

JA -3

CPU CAN TAKE PC VALUE: 101
ADD OFFSET VALUE : +3
104

104 BECOMES THE NEW PC VALUE
104 IS NOW ADDRESS OF NEXT INSTRUCTION TO BE EXECUTED