# Example

```
string1 resb 20
string2 resb 20
strLen resd 1

mov eax, 3
mov ebx, 0
mov ecx, string1
mov edx, 20
int 80h
mov [strLen], eax
mov ecx, [strLen]
mov esi, string1
mov edi, string2
cld
rep movsb
```

# III. STRUCTURED ASSEMBLY LANGUAGE PROGRAMMING TECHNIQUES

## Structured Data Types

# Outline

1. Arrays
2. Strings
3. Structures/Records
4. Sets

# Structures/Records

- collection of data with different types
- contiguous bytes of memory divided according to data type used by user
- We need to know the following:
  - the size of the whole structure
  - the size of each field
  - the starting address of each field

# Implementing Structures

```
struct  student {
      char name[10];
      int age;
      int score;
};
struct student x;
```

```
name      =    10
age       =     2
score     =     2
                14 bytes
```

Define structure size:

```
student equ 14
```

Define starting byte of each field:

```
name  equ    0
age   equ   10
score equ   12
```

Reserve space for structure:

```
x resb student
```

# Implementing Structures

Define structure size:

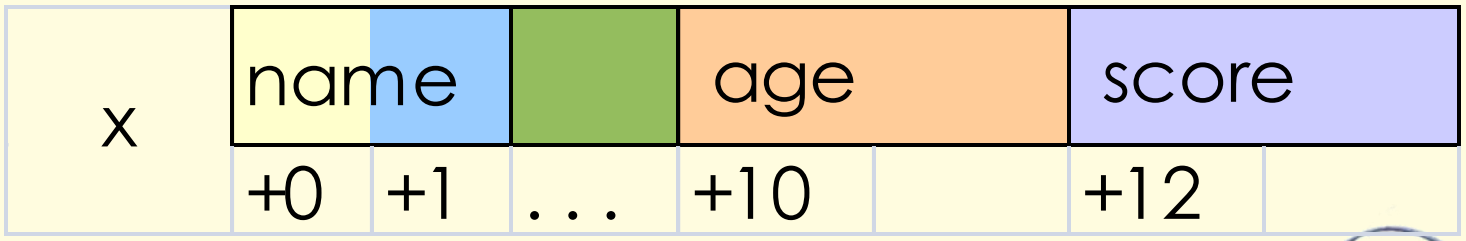
```
student equ 14
```

Define starting byte of each field:

```
name    equ 0
age     equ 10
score   equ 12
```

Reserve space for structure:

```
x resb student
```

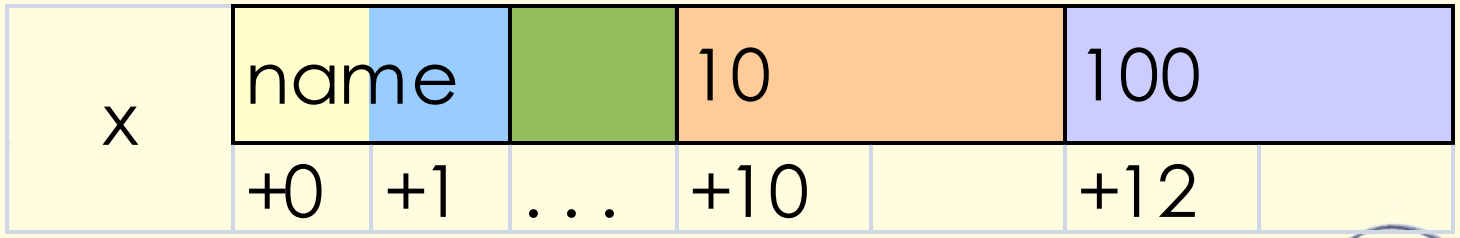


# Accessing Fields

- To access a specific field, just identify the address of that field by specifying its offset from the base address.
- x.age = 10;

  mov word[x+age], 10
- x.score = 100;

  mov word[x+score], 100



# Array of Structures

```
struct  student {
      char name[10];
      int age;
      int score;
};
struct student x[5];
```

```
name      =    10
age       =     2
score     =     2
                14 bytes
```

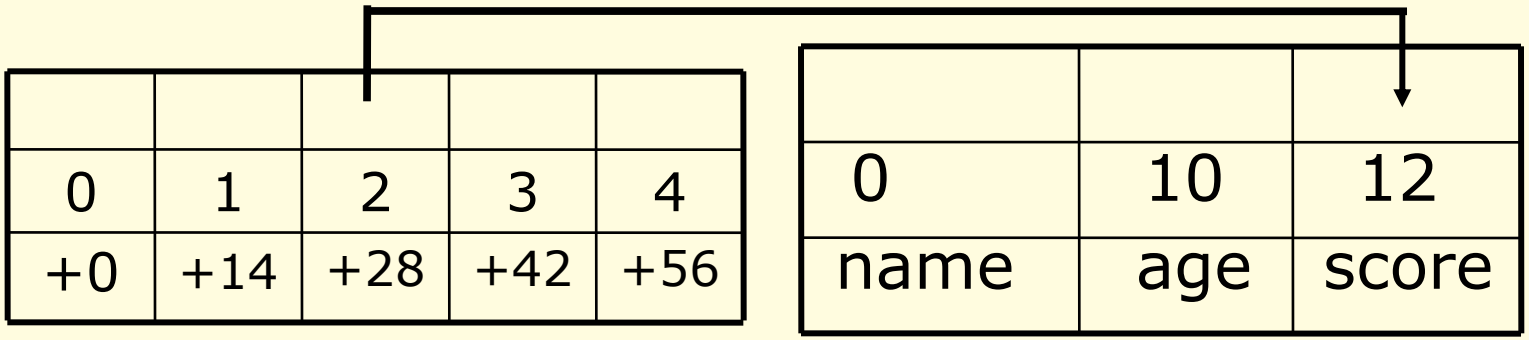
```
array_size    equ      5
student       equ     14
name          equ      0
age           equ     10
score         equ     12
```

```
x  resb array_size*student
```

# Array of Structures

- specify array cell to use (i*size)
- specify field to use

- x[2].score = 100;
- mov word[x+student*2+score], 100



# Array of Structures with Array Field

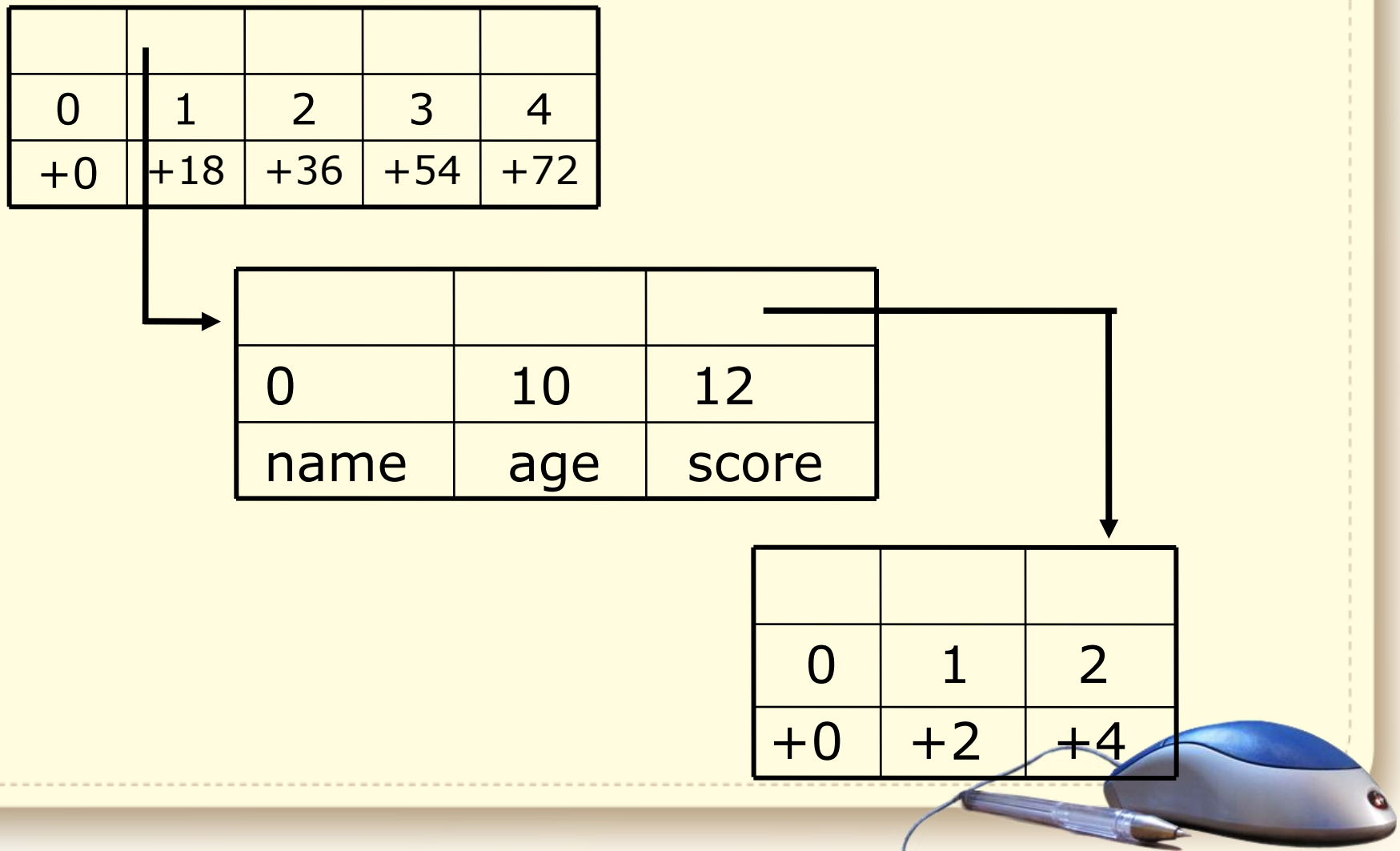
```
struct  student {
      char name[10];
      int age;
      int scores[3];
};
struct student x[5];
```

```
name      =    10
age       =     2
score     =     6
                18 bytes
```

```
array_size    equ      5
integer       equ      2
student       equ     18
name          equ      0
age           equ     10
scores        equ     12

x  resb array_size*student
```

# Array of Structures with Array Field



# Array of Structures with Array Field

```
x[1].scores[2] = 100;                    ; C equivalent


mov word[x+student+scores+integer*2], 100
```