Structures and Records.

Structures. Description of Structure Picture.

Determination (declaring) data with a structure type. Methods of work with structures.

Records. Description of Records. Determination of record exemplar.

Work with records.

For what purpose are structured data types included in assembly language?

A structured data type is one in which each data item is a collection of other data items. In a structured data type, the entire collection uses a single identifier (name). The purpose of structured data types is to group related data of various types for convenient access using the same identifier.

What structured data types do MASM and TASM support?

MASM and TASM support the following structured data types:
arrays;
structures;
associations;
records.

Def.1 Structure is a type of data, which consists of fixed number of elements of different characteristics.

Structures can be considered as pictures with descriptions of data formats, which may be patched (наложены) on different areas of memory with a purpose – to have a possibility for addressing to fields of these areas with help of mnemonic names, which are determined in the structure description.

Structures are especially useful in cases, when it is necessary to address for memory areas, which haven't been included in program's segments (i.e. to such fields, which are not possible to describe with help of symbolic names). Structures are also used in such situations, when a program includes complex collections of data, which repeat many times and have different meanings.

A data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data.

For using structures in the program, it is necessary:

- To set ***a picture*** of the structure, i.e. to determine a new type of data, which will be used in future for determination of variables of this type;
- To determine an exemplar of the structure, i.e. to initialize a certain variable with the beforehand determined (with help of the picture) structure;
- To organize access (addressing) to this variable.

Structure may be described only once, but it can be determined many times.

The description of a structure has got the following syntax:

***name_of_structure* STRUC**

<description of fields>

***name_of_structure* ENDS**

```
workers STRUC    ; information concerning some worker
name      DB   30 DUP (" "); surname, name, patron
position  DB   30 DUP (" ")
age       DW   ?
salary    DD   ?
birthdate DB   30 DUP(" ")
workers ENDS
```

```
data     segment
Sotr1  workers <"Kozlov Susik  Musikovich", , 'artist',99,1000000,'01.10.1917'>
Sotr2 workers<"Frackinbok Matilda Karlsovna",'dancer',18,'90000000','11.12.2000'>
Sotr3  workers <>; here all meanings are on default
     data  ends
```

address_expression.name_of_the_structure

sotr2.salary

[bx].age

Records.

Def. Record is a structured type of data, which consists of fixed number of elements of the size from one up to a number of bits.

The using of records in a program are organized in three stages:

- Setting a picture of record, i.e. determination of bits fields, their lengths, and, if it's necessary, initialization of the fields;
- Determination of record exemplar. As it is made for structures, this stage means initialization of certain variable by type of beforehand determined record with help of picture.
- Organization of addressing to the record's elements.

The using of records in a program are organized in three stages:

- Determination of record exemplar. As it is made for structures, this stage means initialization of certain variable by type of beforehand determined record with help of picture.
- Organization of addressing to the record's elements.
- Setting a picture of record, i.e. determination of bits fields, their lengths, and, if it's necessary, initialization of the fields;

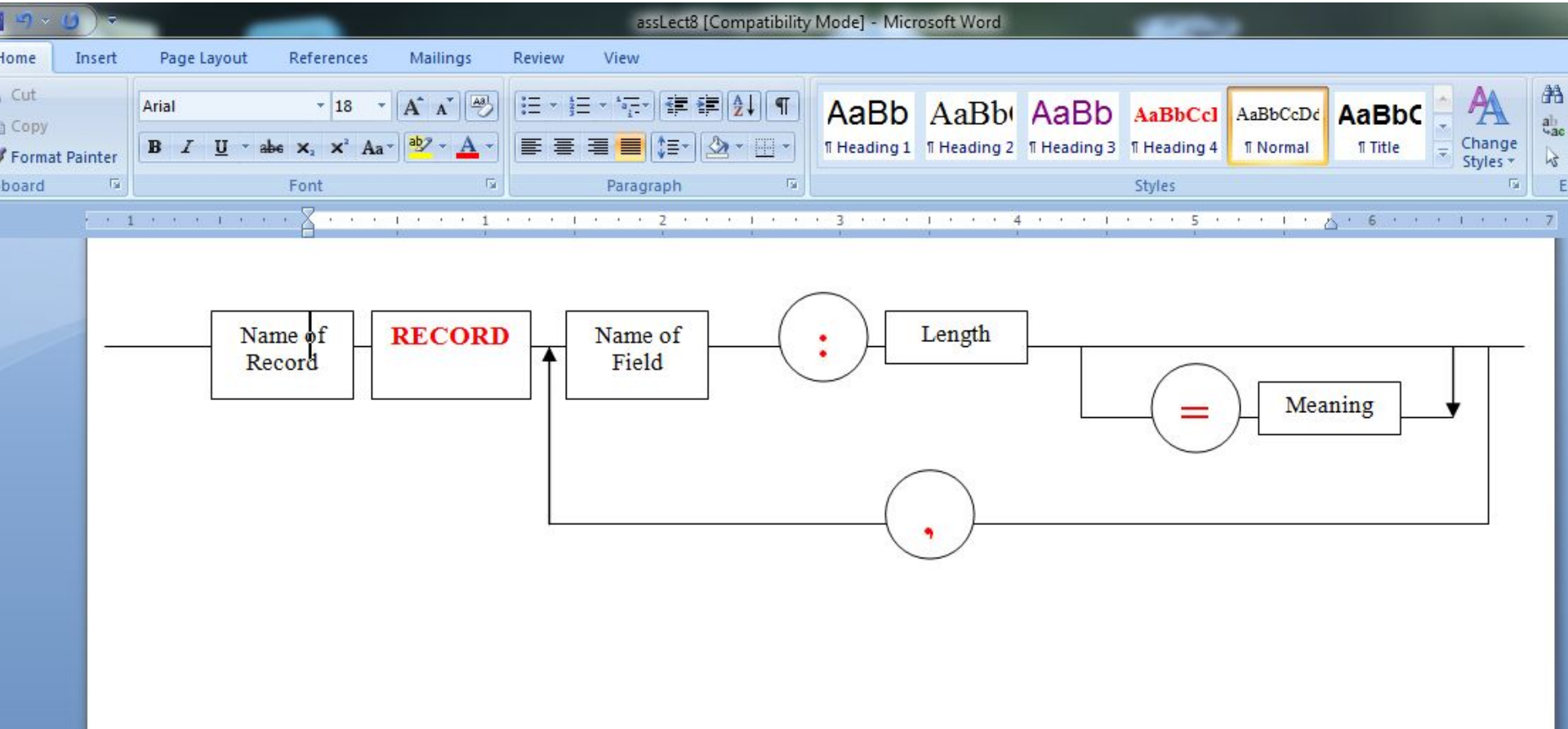
Record Description.

The description of picture has got the following syntax:

name_of_record* RECORD *<description of elements>

here

<description of elements> is a sequence of different elements descriptions according the syntax agram



.....

name_1 RECORD fld_1:5=8, fld_2:6=4, fld_3:2, fld_4:6=6

.....

name_10 name_1?; there is no necessity to initialize fields

.....

name_1 RECORD fld_1:5=8, fld_2:1=4, fld_3:2, fld_4:1=6

.....

name_10 name_1 <, ,7, >; before the third position there are “, ,”

or

.....

name_1 RECORD fld_1:5=8, fld_2:1=4, fld_3:2, fld_4:1=6

.....

name_10 name_1{fld_3=7}

How is the Record bit-field addressed in assembler?

Work with Records.

It is not possible to use usual mechanisms for addressing to record's elements

- to each element of record the translator (assembly) puts in correspondence (assigns) a numeric value, which is equal to a number of shifts to the right;
- the shift to the right is performed with help of instruction ***shr*** ;
- with help of operator ***width*** it is possible to determine a size of element (or the record as a whole) in bits. There are the next variants of using this operator
- **width** *name_of_record's_element* – the meaning of this operator will be a size of the element in bits;
- **width** *name_of_record's_examplar*

  or
- **width** *name_of_record's_type*
- the Assembly has an operator **mask,** which allows to localize bits of record's element;
- all operations, concerned with transformation record's elements are executed with help of logical instructions.

Sequence of actions, which are to be performed for processing a given element of record:

- **To put the given element into the register;**
- **By using operator mask, to obtain a bit mask;**
- **To localize bits in the register with help of instruction and;**
- **To shift bits of element to the junior digit places of the register.**