# Lab 14

# Topics:

1. Structures
2. Directives

# Structures:

Structure is a template or pattern given to a logically related group of variables. The variables in a structure are called fields. Program statements can access the structure as a single entity, or they can access individual fields. Structures often contain fields of different types.

A structure is defined using the STRUCT and ENDS directives. Inside the structure, fields are defined using the same syntax as for ordinary variables. Structures can contain virtually any number of fields:

*Name* STRUCT

*field-declarations*

*Name* ENDS

Example:

A structure defined for the x and y coordinates of the console is defined as follows:

*Coord structs*

*x\_cor word ?*

*y\_cor word ?*

*Coord ends*

The fields in the structure can be defined as ordinary variables are declared in the code.

The structure variables can be defined under the data directive as follows:

*.data*

*point1 Coord <5, 10> ; x\_cor = 5, y\_cor = 10*

*point2 Coord <20> ; x\_cor = 20, y\_cor = ?*

*point3 Coord < > ; x\_cor = ?, y\_cor = ?*

The fields or variables of a structures are declared inside the structure, while the value can be assigned to the fields during run time. Example shows how the fields are accessed.

*.code*

*mov point1.x\_cor, dx ; moves the value of dx in the x\_cor variable of point1*

*mov point2.y\_cor, ax ; moves the value of ax in the y\_cor variable of point2*

All the functions that we have previously used are valid for the fields of the structures.

# Directives:

Directives are instructions to the assembler rather than the CPU. They provide information to the assembler about how to process the source code. Directives do not generate machine code; instead, they guide the assembler in organizing the program, allocating memory, defining constants, and more.

Here are some commonly already you used directives in your program for example:

.Model

.Code

.Data

## Conditional Directives:

The conditional assembly directives can be used in conjunction in macro to make them more flexible.

## IF, ELSE and ENDIF directives:

The IF directive must be followed by a constant Boolean expression. The expression can contains integer constants, symbolic constants or constant macro argument but it cannot contains registers and variable names. Syntax format uses

IF expression

Statement-list

ENDIF

Another format uses IF, ELSE and ENDIF directives

IF expression

Statement-list

ELSE

Statement-list

ENDIF

IF expression 🡪this lines permits assembly if the value of expression is true(nonzero).possible relational operator are LT,GT,EQ,NE,LE,GE.

ENDIF 🡪Ends a blocks that was begun using the conditional assembly directives.

ELSE🡪Assemble all statements up to ENDIF if the condition specified by previous conditional directive if false.

## Boolean Expression:

The assembler permits the following relational operators is to be used as constant Boolean expression.

* LT means less than
* GT means greater than
* EQ means Equal to
* NE means Not equal to
* LE means less than or equal to
* GE means greater than or equal to

## While Directive:

The WHILE directive repeats a statement of blocks as long as particular constant expression is true. The syntax is

WHILE constant expression

Statements

ENDM

The WHILE directive repeats a statement block based on a Boolean value.

### **Lab Tasks:**

1. **Student Record:** Create a program that manages student records using a struct. Each student's record can include information such as ID, name, age, and grades. Allow the user to input and display student records.
2. Write an assembly program to check whether the number inputted in prime or not. (**You must use conditional directives for this program.)**
3. Write a program that prints the numbers from 1 to 100. However, for multiples of 3, print "Fizz" instead of the number, and for the multiples of 5, print "Buzz." For numbers that are multiples of both 3 and 5, print "FizzBuzz.”. (**You must use conditional directives for this program.)**

**Submission Instructions:**

* Write your name, roll no and section on top of your code file
* Do all of your code work in procedures and just call that procedure in main to execute the code.

Eg:

Q1 proc

; code

ret

Q1 endP

* **Self-Evaluation: (bonus 2 marks)**

You are to self-evaluate all the questions you do. Write a one-liner comment for each question on top of code file below your name.

Eg. Q1, all done, works completely fine for all use cases

Q2, partially done, exception thrown error

**You must be honest in self-evaluating yourself.**

**If during evaluation you are found being dishonest, your lab will be marked zero.**

* Submit only one .asm file (Format: i22-1234\_LAB14.asm