Alexandria University,

Faculty of Engineering,

Computer and Systems Engineering Dept.

Systems Programming

CS222

*Final Project*

SIC/XE Assembler

Phase2

Names:

1. Ziad Taha Ali Mekawy. (18)
2. Ziad Hisham Ali. (19)
3. Mohamed Zakaria Mohamed (39)
4. Mostafa Mohamed Mahmoud (50)
5. Youssef Ahmed Sayed Gabriel. (61)
6. Problem Statement:

Pass 2 Assembler, use pass 1 that already implemented to compile the program to OpCode and generate the ObjectFile .

1. Requirements Specification:

* **Make three files.txt, one for the input that contain the Assembly Code & another will be wrote on it the list file Assembler and the Opcode & the last one will be wrote on it the generated ObjectFile**
* **Enter the name of the input File**
* **Output will be printed in the cmd and wrote on output file**

1. Design:

* **Mnemonic “Class”**

Used to check the syntax of the Assembly language & To get Opcode

* **String getOpCode(string)** used to get opcode of operands
* **Int getLength(string)** used to get length of the statement
* **String getAttribute(string)** used to get attribute
* **Bool isFound(string)** check if statement exist or not
* **Object\_Code “Class”**

Used to get the opCode of each line in the program

* **Int operation(a, b, operator)** used to make operations in two numbers in hexadecimal system.
* **bool checkIfExp (attribute)** used to check if the given attribute is an expression or not.
* **Bool checkAandR (expresion1,expression2,operator)** used to check if the expression is valid or not .
* **String getRegisterNum (register)** used to set the number of the given register.
* **Bool calcDisplacement (Address, nextAdress,**

**Attribute, bool isPlus)** used to calculate the displacement of the currentLine with the PC relative or BASE relative “Format3” or with No One of the both “Format4”.

* **Vector<String> getErrorMes ()** used to declare the error if it is.
* **String evaluateLine (line, nextLine)** used to get the OpCode of the currentLine through calling the previous functions in definite sequence.

* **Main “Class”**

Implement the Algorithm

* **String read(fileName)** used to read the input file, read it line by line & push it in vector of strings.
* **Bool IsThere(label, symbolTable)** usedto check if the given label valid or not.
* **Int CheckLast(Attribute, operand)** used to check if the given attribute and operand are valid to gether or not through REGEX.
* **String returnCount(label, symbolTable)** used to get the address of the given label.
* **Void errorsM (counter, pass1, lineNumber, start, currentLine, bool check, error)** used to print out the detected errors.
* **bool checkLableSymbol (symbolTable, attribute, value, absolute)** used check if the given attribute present in the symbolTable that generated in pass1.
* **Void exeprssionForReserving (a, b, operand, symbolTable, pass1, lineNumber, start, currentLine, bool check, bool word)** used to save the given expression in the symbolTable after checking it.
* **Int main()** iterate on the vector of strings filled through the input file, then using the REGEX to validate the line form , then check if the syntax is valid using ***isThere “method” ,*** then check the operand field in the line using ***checkLast “method”*** , then get the address of the line using ***returnCount “method”*** & Finally if there’s anything wrong in the previous stages Error will be declared and wrote in the output. “Pass1”

Check if the given symbol presence in the generated symbolTable if not declare an Error, if there’s an expression check it using ***checkAandR “method”*** if it’s valid reserve it on the symbolTable using ***exeprssionForReserving “method”*** and if not declare an Error using ***errorsM “method” ,*** then calculate the opCode of each line in pass1 using **Object\_Code “Class”** to get the opCode of each line in pass1 & Eventually generate the ObjectFile through reformat the opCode in specific format and write it on Object\_File.txt “Pass2”

* **Errors**
* **Check if the input file is valid**
* **Check if the given line is in a right form through REGEX**
* **Check if the statements are written in a right syntax**
* **Check if the operation & operand filed are right to gether**
* **Check if the format written in a right syntax**
* **Check if there’s no duplicates in the labels names**
* **Check if the expression written in a right form**
* **Check if the given expression is valid or not**
* **Check if the symbolTable doesn’t have the target symbol**
* **Check if the address after the displacement doesn’t exceed the limit of Base relative (2047)**

1. Main Data Structures:

* Vector

Used to store the input file as strings and to write opCode & ObjectFile.

* Array

Used to with size 3, as the vector is used as a vector of array & for the SymbolTable.

* Pair

Used in the map as a value to store Opcode & length.

* Map

Used to store the operations, Operands, length & Opcode.

* StringStream

Used for easily conversion from String to Int.

1. Algorithms :

* **Read input file and store it in vector**
* **Check if the input form is right**
* **If there’s any error in the checking throw it**
* **Calculate the address of each line**
* **Get the Opcode of each operation**
* **Get all defined labels and their address & states**
* **Print the output (Addresses & SymbolTable)**
* **Get the lines that generated from pass1**
* **Get line by line and check first if the generated symbolTable have the targetSymbol**
* **If there’s expression, first check its format then if it’s valid ,solve it and store in the symbolTable**
* **Get the opCode of each line with el values of each register (0 or 1)**
* **Generate the objectFile of the program**
* **Print the output (OpCode & ObjectFile & Errors)**

1. Assumptions:

* **Free formatted is supported, there’s no restriction to begin at a given position “Bonus”**
* **In the objectFile the length of the opCode in the text record is “1D” 10 lines as a maximum and decreasing the length if the address doesn’t exist.**

1. Sample Runs:





























