|  |  |
| --- | --- |
| **Ex No: 4** | **Single Pass Assembler For SIC** |
| **Date of Exercise** | **20/01/2022** |

|  |
| --- |
| **AIM** |
| To simulate the concept of single pass assembler for simplified instructional computer. |
| **DESCRIPTION** |
| An assembler is a program that accepts as input an assembly language program (source) and produces its machine language equivalent (object code) along with the information for the loader.      **Figure 1**. Executable program generation from an assembly source code  Advantages of coding in assembly language are:   * Provides more control over handling particular hardware components * May generate smaller, more compact executable modules * Often results in faster execution |
| **ALGORITHM** |
| 1. Open and Read the input file 2. If the input line has the opcode START do the following   2.1 Find if there is any operand field after START initialize the LC to the operand  Value.   * 1. Otherwise if there is no value in the operand field then LC is set to 0.  1. Write the input line to the intermediate file. 2. Do the following steps until the opcode is END.   4.1 Check the Symbol table, if the symbol isnot available then enter that symbol into  the SYMTAB, along with the memory address in which it is stored.Otherwise, the  error message will be displayed.   * 1. If there is a opcode      1. 1 If opcode is present in the OPTAB, then increment the LC by 3 and   Start writing the location counter, opcode and operand fields of the  corresponding statement to the output file, along with the object code.   * + 1. If opcode isWORD then increment LC by 3     2. If opcode is BYTE then increment LC by 1     3. If opcode is RESW then increment LC by the integer equivalentof the operand value \* 3     4. If opcode is RESB then increment LC by the integer equivalentof the operand value     5. If there is no symbol/label in the operand field, then the operandaddress is assigned as zero and it is assembled with the object code of the instruction     6. Write the processed lines in the intermediate file along with their along with the location counters.  1. To find the length of the program, Subtract the starting address of the program from the LC. 2. Close all the files. |
| **SAMPLE INPUT & OUTPUT** |
| **Input.txt:**   |  |  |  | | --- | --- | --- | | COPY | START | 1000 | |  | LDA | ALPHA | |  | STA | BETA | | ALPHA | RESW | 1 | | BETA | RESW | 1 | |  | END |  |   **optab.txt**   |  |  | | --- | --- | | STA | 23 | | LDA | 00 |   **Symtab.txt**   |  |  | | --- | --- | | ALPHA | 1006 | | BETA | 1009 |   **Result.txt**  H^COPY^1000^0c  T^001000^06^001006^231009  E^001000 |
| **QUESTION SET** |
| 1. Write a program to simulate single one pass assembler for simplified instructional computer. |

**PROGRAM:**

op\_table = {}

labels = []

commands = []

values = []

memory\_locations = []

op\_code = []

object\_code = []

"""Reading content in text file "symtab.txt" and printing the content"""

sym\_table = {sym.split()[0]: sym.split()[1] for sym in open("/content/drive/MyDrive/Exercise 4/symtab.txt").readlines()}

print(sym\_table)

with open("/content/drive/MyDrive/Exercise 4/opentab.txt") as optab:

optable = optab.readlines()

for line in optable:

op\_table[line.split()[0]] = line.split()[1]

with open("/content/drive/MyDrive/Exercise 4/input .txt") as input\_:

input\_data = input\_.readlines()

for line in input\_data:

if "START" in line: # Detect starting point of the program

start = line.split()[2]

memory\_locations.append("")

x = int(start, 16)

else:

if len(line.split()) == 3:

memory\_locations.append((sym\_table.get(line.split()[0])))

else:

memory\_locations.append(str(hex(x)[2:]))

x += 3

for line in input\_data:

line = line.split()

if len(line) == 3:

labels.append(line[0])

commands.append(line[1])

values.append(line[2])

elif len(line) == 2:

labels.append("")

commands.append(line[0])

values.append(line[1])

else:

commands.append(line[0])

for command in commands:

if command == "START":

continue

if command in op\_table:

x = str(op\_table[command])

y = str(memory\_locations[labels.index(values[commands.index(command)])])

object\_code.append(f"^{x + y}")

ob = "".join(object\_code)

x = str(hex(len(object\_code) \* 3)[1:])

if "x" in x:

x = x.replace("x", "0")

with open("/content/drive/MyDrive/Exercise 4/result.txt", "w+") as result:

result.write(f"H^{labels[0]}^{memory\_locations[1]}^{memory\_locations[-1][2:]}\n")

result.write(f"T^{memory\_locations[1]}^{x}{ob}\n")

result.write(f"E^00{memory\_locations[1]}\n")

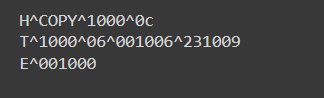
"""Printing the content of texfile "result.txt""""

with open("/content/drive/MyDrive/Exercise 4/result.txt") as f:

output=f.read()

print(output)

**OUTPUT SCREENSHOT:**



**RESULT:**

Thus the program ran successfully and the output was verified.