**Цель работы:** «Перевод ОПЗ исходного выражения в текст на выходном языке. Генерация машинного кода»

Перевод ОПЗ в текст на выходном (машинном) языке представляет собой следующий этап трансляции исходной программы в машинные коды. Для реализации этой процедуры также используется автомат с магазинной памятью (МП-автомат). Эту процедуру можно схематично представить следующим образом:

**Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание**

Чтение символов операций из ОПЗ инициирует семантические процедуры, которые генерируют соответствующие заготовки машинного кода. Так же обрабатываются собранные в стеке операнды данной. Например, в случае обработки переменных и констант извлекается соответствующая им информация из таблиц идентификаторов и констант, которая используется для образования правильных адресных частей соответствующих машинных команд.

**Код программы:**

def rpn\_to\_infix(expression):

operations = {"=", "+=", "-=", "\*=", "+", "-", "\*", "/", "\*\*"}

stack = []

infix = []

func = 0

ifka = 0

cic = 0

totalcic = 0

for token in expression.split():

if token.isalnum() and not 'АЭМ' in token and not 'Ф' in token and not 'УПЛ' in token and not 'БП' in token and not 'МЦ' in token:

if not "М" in token:

stack.append(token)

elif 'АЭМ' in token:

r = int(token[0])

arr = []

newarr = []

while r != 0:

arr.append(stack.pop())

r -= 1

arr = arr[::-1]

for i in range(len(arr)):

if i == 0:

newarr.append(arr[i])

newarr.append('[')

else:

newarr.append(arr[i])

if i + 1 < len(arr):

newarr.append(',')

newarr.append(']')

stack.append(''.join(newarr))

elif 'Ф' in token:

func = 1

r = int(token[0])

arr = []

newarr = []

while r != 0:

arr.append(stack.pop())

r -= 1

arr = arr[::-1]

for i in range(len(arr)):

if i == 0:

newarr.append(arr[i])

newarr.append('(')

else:

newarr.append(arr[i])

if i + 1 < len(arr):

newarr.append(',')

newarr.append(')')

stack.append(''.join(newarr))

elif token == 'УПЛ':

infix = list("if " + infix[0] + ":\n ")

ifka = 1

elif 'БП' in token:

infix.append("\nelse:\n ")

ifka = 1

elif 'МЦ1УПЛ' in token:

infix.append("while " + infix.pop() + ":\n ")

cic = 1

totalcic = 1

elif len(stack) != 0:

if len(stack) == 1:

infix.append(stack.pop())

break

else:

b = stack.pop()

a = stack.pop()

op = " ".join([a, token, b])

if func == 1:

infix.append(op)

func -= 1

if ifka == 1:

stack.append(op)

ifka = 0

if cic == 1:

infix.append(op)

cic = 0

else:

infix.append("(" + op + ")")

if token in operations:

infix.append('\n')

if token in operations and totalcic == 1:

infix.append(' ')

infix = ''.join(infix)

if not 'Ф' in expression:

infix = infix.replace("(", "")

infix = infix.replace(")", "")

return infix

**Примеры работы программы:**

Пример №1: Обработка элементов с индексами

Входная строка: s i j 3АЭМ 10 +=

Результат работы программы: s[i,j] += 10

Пример №2: Обработка полного условного оператора

Входная строка: a b > М1 УПЛ a x y - += M2БП М1: a b += М2:

Результат работы программы:

if a > b:

a += x - y

else:

a += b

Пример №3: Обработка элементов внутри цикла

Входная строка: a 10 = b 20 = a b < МЦ1УПЛ b 5 -= a 1 += МЦ1:

Результат работы программы:

a = 10

b = 20

while a < b:

b -= 5

a += 1

**Вывод:** в ходе данной лабораторной работы было осуществлен перевод обратной польской записи, полученной в лабораторной работе №2, в код на выходном языке python.