УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №1

по предмету

Компиляторные технологии

Вариант 12

Выполнил:

Крутько А. А.

Проверила:

Шостак Е. В.

Группа 251004

Минск 2023

Задание:

Двоичное, восьмеричное или 16-чное число в синтаксисе FASM (постфиксная форма записи). 16-чное число должно начинаться с цифры от 0 до 9.

**Регулярное выражение**

Константы:

DigBin = [0-1];

DigOx = [2-7];

DigHex = [8-9];

LetHex = [AaC-Fc-f];

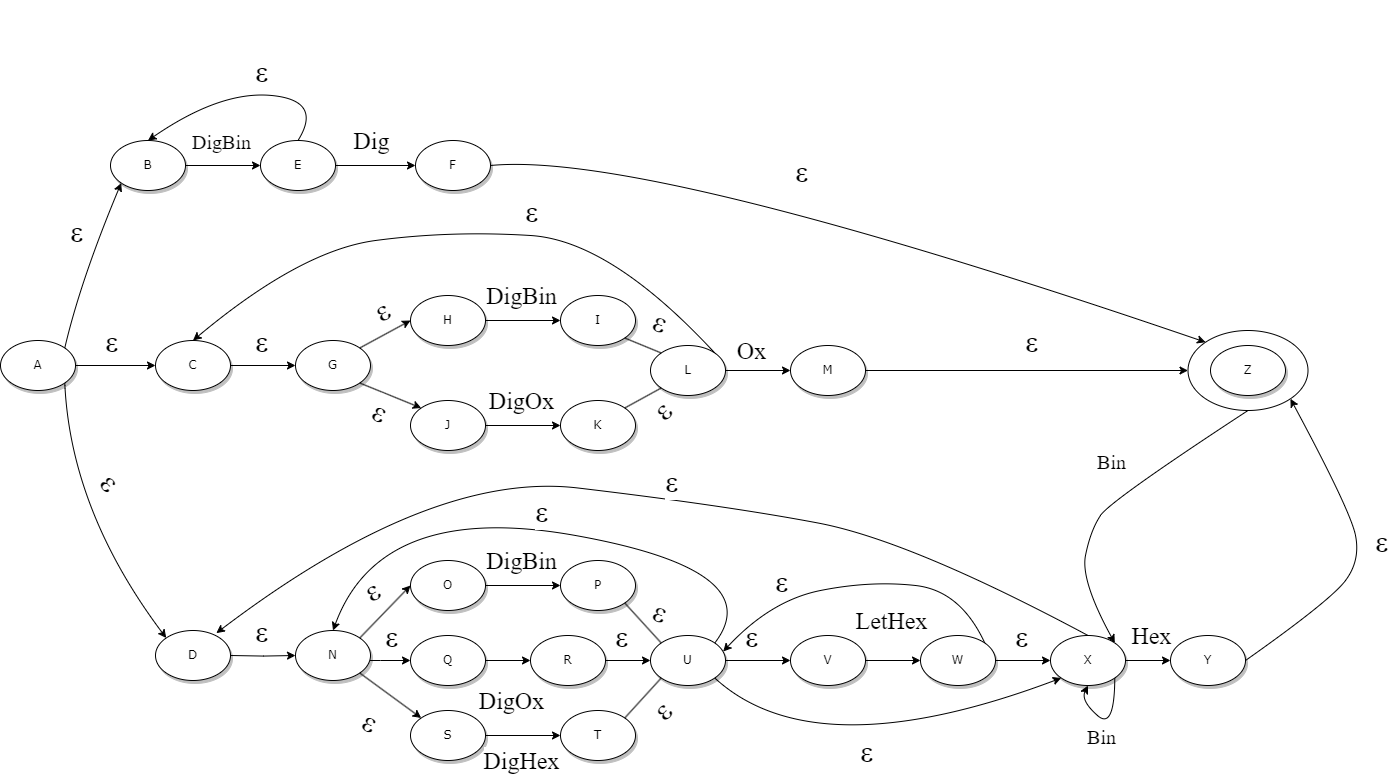
Bin = [Bb];

Ox = [oO];

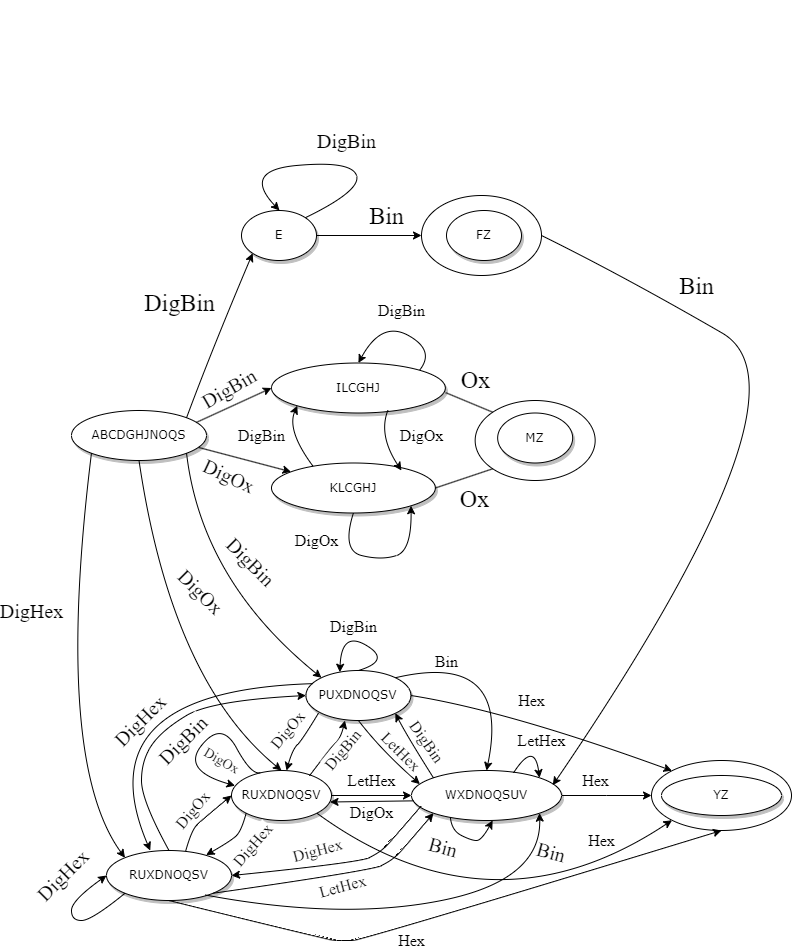
Hex = [Hh];

Number = (DigBin+ Bin) | ((DigBin | DigOx)+ Ox) | (((DigBin | DigOx | DigHex)+ (LetHex)\*)+ Hex);

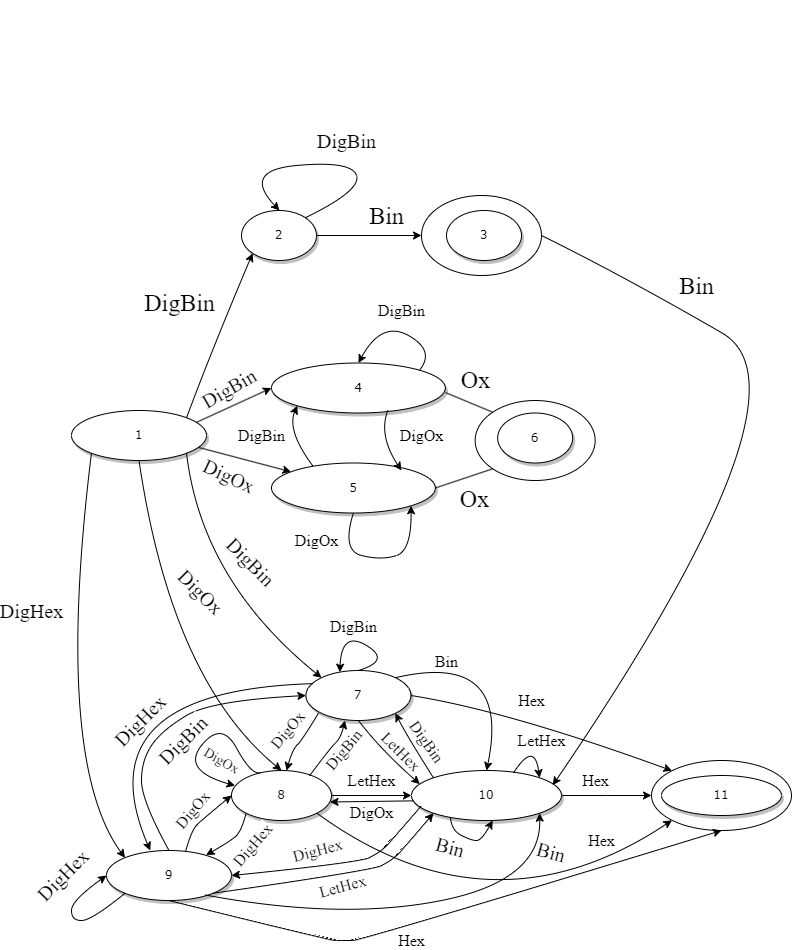
**НКА (для регулярного выражения)**



**ДКА (для регулярного выражения)**

****

**ДКА(готовый к программной реализации)**



**Таблица переходов**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Error** | **DigBin** | **DigOx** | **DigHex** | **Bin** | **Ox** | **Hex** | **LetHex** |
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 2 | 5 | 9 | 0 | 0 | 0 | 0 |
| **2** | 0 | 2 | 5 | 9 | 3 | 6 | 11 | 10 |
| **3** | 0 | 7 | 8 | 9 | 10 | 0 | 11 | 10 |
| **4** | 0 | 4 | 5 | 9 | 10 | 6 | 11 | 10 |
| **5** | 0 | 4 | 5 | 9 | 10 | 6 | 11 | 10 |
| **6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **7** | 0 | 7 | 8 | 9 | 10 | 0 | 11 | 10 |
| **8** | 0 | 7 | 8 | 9 | 10 | 0 | 11 | 10 |
| **9** | 0 | 7 | 8 | 9 | 10 | 0 | 11 | 10 |
| **10** | 0 | 7 | 8 | 9 | 10 | 0 | 11 | 10 |
| **11** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Код программы, реализующий работы ДКА**

#include <stdio.h>

#include "string.h"

#include "windows.h"

#define StateNumbers 12

#define CharType 8

typedef enum {

Error = 0,

DigBin,

DigOx,

DigHex,

Bin,

Ox,

Hex,

LetHex,

} State;

typedef enum {

S\_Unknown = 0,

S\_Bin,

S\_Ox,

S\_Hex

} Result;

const Result IsFinalState[StateNumbers] = {S\_Unknown, S\_Unknown, S\_Unknown, S\_Bin, S\_Unknown, S\_Unknown, S\_Ox, S\_Unknown, S\_Unknown, S\_Unknown, S\_Unknown, S\_Hex};

const int Transitions[StateNumbers][CharType] =

{

{0, 0, 0, 0, 0, 0, 0, 0},

{0, 2, 5, 9, 0, 0, 0, 0},

{0, 2, 5, 9, 3, 6, 11, 10},

{0, 7, 8, 9, 10, 0, 11, 10},

{0, 4, 5, 9, 10, 6, 11, 10},

{0, 4, 5, 9, 10, 6, 11, 10},

{0, 0, 0, 0, 0, 0, 0, 0},

{0, 7, 8, 9, 10, 0, 11, 10},

{0, 7, 8, 9, 10, 0, 11, 10},

{0, 7, 8, 9, 10, 0, 11, 10},

{0, 7, 8, 9, 10, 0, 11, 10},

{0, 0, 0, 0, 0, 0, 0, 0}

};

State getCharState(char c) {

switch(c) {

case '0'...'1':

return DigBin;

case '2'...'7':

return DigOx;

case '8'...'9':

return DigHex;

case 'b':

case 'B':

return Bin;

case 'o':

case 'O':

return Ox;

case 'h':

case 'H':

return Hex;

case 'A':

case 'C'...'F':

case 'a':

case 'c'...'f':

return LetHex;

default:

return Error;

}

}

Result CheckString(const char\* s) {

int state = 1;

for (int i = 0; s[i]; i++) {

State result = getCharState(s[i]);

state = Transitions[state][result];

}

return IsFinalState[state];

}

void findInStr(char\* str) {

int state = 1;

int startPos = 0;

int middlePos = 0;

Result prev\_State = 0;

for (int i = 0; str[i]; i++) {

State result = getCharState(str[i]);

state = Transitions[state][result];

if (state == 0) {

state = 1;

startPos = i + 1;

middlePos = i + 1;

prev\_State = 0;

}

if (IsFinalState[state]) {

for (int index = startPos; index <= i; index++)

putc(str[index], stdout);

putc('\n', stdout);

if (IsFinalState[state] == S\_Hex && prev\_State == S\_Bin && middlePos != i) {

for (int index = middlePos; index <= i; index++) {

putc(str[index], stdout);

}

putc('\n', stdout);

}

prev\_State = IsFinalState[state];

if (prev\_State != S\_Bin) {

startPos = i + 1;

state = 1;

} else {

middlePos = i + 1;

}

}

}

}

int main(int argc, char\* argv[]) {

SetConsoleOutputCP(CP\_UTF8);

SetConsoleCP(CP\_UTF8);

printf("Задание: \n");

printf("Двоичное, восьмеричное или 16-чное число в синтаксисе FASM (постфиксная форма записи).\n16-чное число должно начинаться с цифры от 0 до 9.\n");

printf("Константы:\n");

printf("DigBin = [0-1]\n");

printf("DigOx = [2-7]\n");

printf("DigHex = [8-9]\n");

printf("LetHex = [AaC-Fc-f]\n");

printf("Bin = [Bb]\n");

printf("Ox = [oO]\n");

printf("Hex = [Hh]\n");

printf("Регулярное выражение:\n");

printf("(DigBin+ Bin) | ((DigBin | DigOx)+ Ox) | (((DigBin | DigOx | DigHex)+ (LetHex)\*)+ Hex)\n");

while (true) {

printf("Введите строку:\n");

char \*str = lineRead();

Result state = CheckString(str);

if (state == S\_Bin) {

printf("It's a binary number.\n");

} else if (state == S\_Ox) {

printf("It's an octadecimal number.\n");

} else if (state == S\_Hex) {

printf("It's a hexadecimal number.\n");

} else {

printf("It's not a number.\n");

}

printf("Введите строку для нахождения всех подстрок:\n");

str = lineRead();

findInStr(str);

}

}