**Федеральное государственное образовательное бюджетное учреждение высшего образования**

**«Финансовый университет при Правительстве Российской Федерации»**

**Колледж информатики и программирования**

ПМ.03 Участие в интеграции программных модулей

Группа: 4ПКС-115

**Отчет по производственной практике**

**(по профилю специальности)**

**Руководитель практики от предприятия**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель практики от колледжа**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Студент**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Оценка** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.2019

2019

Содержание

[Введение 3](#_Toc3494036)

[1 Общая часть 5](#_Toc3494037)

[1.1 Описание структурного подразделения организации 5](#_Toc3494038)

[1.2 Технические, инструментальные и программные средства 5](#_Toc3494039)

[1.3 Процесс выполнения работы 6](#_Toc3494040)

[Заключение 10](#_Toc3494041)

[Список использованных источников 12](#_Toc3494042)

[Приложение А 13](#_Toc3494043)

# Введение

ПАО «НПП Сапфир» является базовым предприятием российской оборонной промышленности. Оно сосредоточило свои усилия на научно-производственном направлении по созданию радиационно-стойких КМОП, КНС и БИС, технология изготовления которых имеет статус Федерального уровня.

ПАО «НПП Сапфир» осуществляет поставки микропроцессоров для нужд Роскосмоса и Минатома Российской Федерации.

Предприятие имеет огромный опыт производства и разработки наукоемкой продукции, применяемой в бортовых системах и вычислительных комплексах баллистических ракет.

Сочетание традиционных и новых направлений в разработках и производстве полупроводниковой электроники, создание законченных изделий и устройств для первоочередных нужд народного хозяйства и обороны страны позволяет НПП «Сапфир» развивать науку и производство России.

При ПАО «НПП Сапфир» на базовой кафедре «Некогерентная оптоэлектроника» проводятся по специальным предметам занятия по специальности 210101 «Микроэлектроника и твердотельная электроника».

Подготовка инженеров проводится в соответствии с действующими Государственными образовательными стандартами.

На производственной базе ПАО «НПП Сапфир» работают множество предприятий малого и среднего бизнеса.

Двадцать из них объединены в созданном в 2005 году Технопарк «Сапфир».

Целью прохождения практики в данной организации является:

* Ознакомление с деятельностью организации;
* Закрепление и развитие практических навыков, общих и профессиональных компетенций в процессе выполнения определенных видов работ, связанных с будущей профессиональной деятельностью.

Задачами практики является:

* Систематизация, закрепление и расширение теоретических и практических знаний, полученных в процессе обучения;
* Проведение анализа и обобщения научно-технической информации по теме выпускной квалификационной работы.

Вопросы, подлежащие изучению.

* Литературные источники по разрабатываемой теме с целью их использования при выполнении выпускной работы;
* Методы исследования и проведения работ, анализа и обработки данных;
* Информационные технологии и технические средства в профессиональной деятельности;
* Программные продукты, относящиеся к профессиональной сфере;
* Критерии выбора, сравнения и оценки эффективности решений по защите информации;
* Требования к оформлению научно-технической документации.

# Общая часть

### Описание структурного подразделения организации

Практика проходила в конструкторско-технологическом отделе ПАО «НПП Сапфир». Основной задачей этого отдела является разработка программ для контроля качества произведённой продукции.

Измерения контроля качества проводятся на специальных установках, на которых микросхемы подвергаются испытаниям в самых различных условиях, полученные в ходе испытаний данные сравнивают с эталонными и вычисляют процент их отклонения. Далее микросхемы отправляют на семидневную токовую тренировку, в этот промежуток времени микросхемы подвергаются нагрузкам в различных температурных условиях, а затем измерения проводятся снова.

Таким образом отбивается продукция высшего качества и определяются недочёты, исправление которых в дальнейшем позволяет совершенствовать будущие микросхемы.

### Технические, инструментальные и программные средства

Практиканту было необходимо разработать «Эмулятор микросхемы». Микросхема представляет собой секцию арифметического 8-разрядного расширителя, управляемого микропрограммно и предназначенного для увеличения скорости выполнения операций умножения сдвигов и нормализации.

Для выполнения поставленной задачи было предоставлено:

* Рабочий компьютер;
* Среда разработки Visual Studio 2017;
* Входные данные для тестирования программы;
* Данные типов штифтов микросхемы;
* Доступ к техническому описанию микросхемы;
* Доступ к временной диаграмме;
* Доступ к структурной схеме;
* Доступ к системе микрокоманд.

### Процесс выполнения работы

В ходе выполнения задания практики была создана программа «Эмулятор микросхемы». Программа позволяет эмулировать работу микросхемы, которая представляет собой секцию арифметического 8-разрядного расширителя, управляемого микропрограммно и предназначенного для увеличения скорости выполнения операций умножения сдвигов и нормализации. Программа написана на языке программирования C# в такой среде разработки как Visual Studio 2017.

Схема классов программы продемонстрирована на рисунке 1.1.

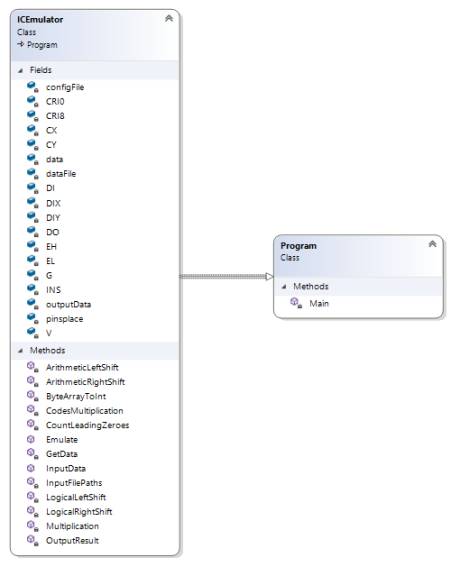


Рисунок 1.1 – Схема классов программы

Программа работает в соответствии с чертежом микросхемы (которая является засекреченной и в данном отчёте не будет продемонстрирована).

Для эмуляции работы микросхемы, в программе, в соответствии с данными о штифтах, на каждый штифт подаётся некоторая последовательность байт, которая в соответствии с чертежом микросхемы проходит выполняет некоторые операции, указанные ниже:

* Подсчёт ведущих нулей;
* Умножение кодов;
* Сдвиг левый логический;
* Сдвиг правый логический;
* Умножение чисел;
* Сдвиг левый арифметический;
* Сдвиг правый арифметический.

После обработки всех входных данных результаты выводятся в специальный файл. На этом работа программы заканчивается.

Результаты работы программы представлены на рисунках 1.2 – 1.5.

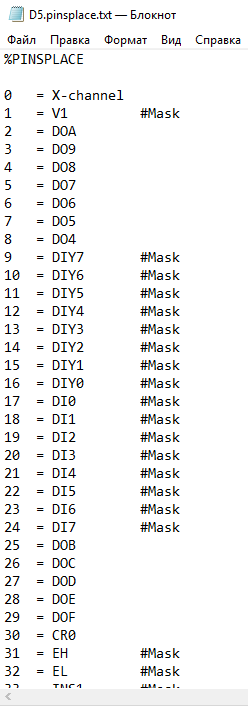


Рисунок 1.2 – Файл с данными о типах штифтов

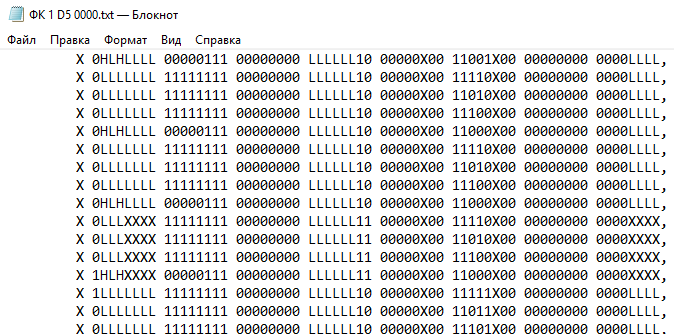


Рисунок 1.3 – Файл с входными данными

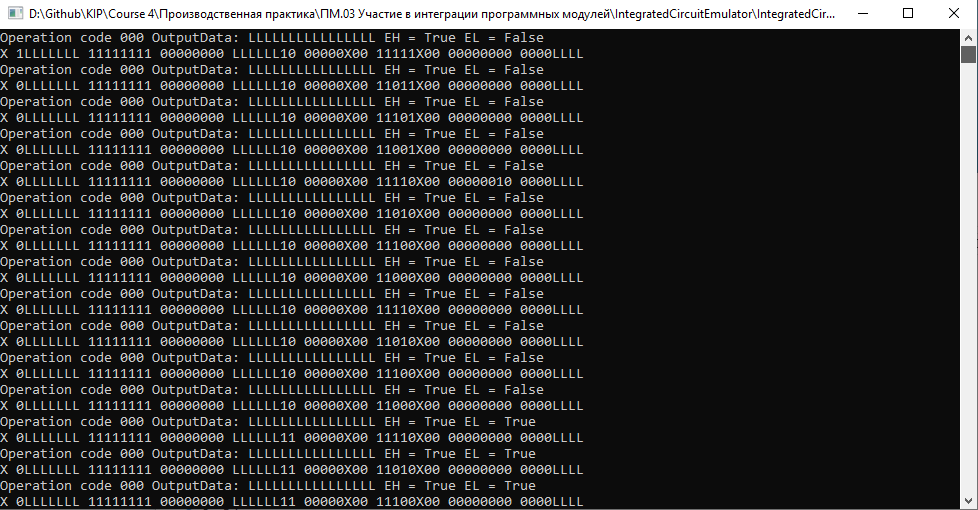


Рисунок 1.4 – Демонстрация работы программы

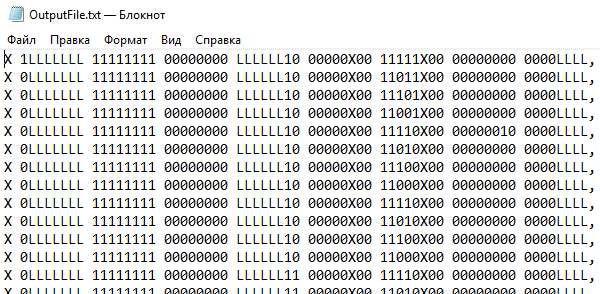


Рисунок 1.5 – Файл с выходными данными

# Заключение

Во время прохождения производственной практики по профессиональному модулю ПМ.03 «Участие в интеграции программных модулей» были изучены:

* Специфика работы, а также технические и программные средства, используемые в конструкторско-технологическом отделе;
* Конструкторская документация и различные стандарты предприятия;
* Информационные технологии и технические средства в профессиональной деятельности;
* Программные продукты, относящиеся к профессиональной сфере;
* Критерии выбора, сравнения и оценки эффективности решений по защите информации;
* Требования к оформлению научно-технической документации.

В ходе прохождения практики была создана программа «Эмулятор микросхемы». Программа написана на языке программирования C# в такой среде разработки как Visual Studio 2017. В процессе написания программы были получены следующие профессиональные компетенции:

* Анализ проектной и технической документации на уровне взаимодействия компонент программного обеспечения;
* Выполнение интеграции модулей в программную систему;
* Выполнение отладки программного продукта с использованием специализированных программных средств;
* Осуществление разработки тестовых наборов и тестовых сценариев;
* Проведение инспектирования компонентов программного продукта на предмет соответствия стандартам кодирования;
* Разработка технологической документации;
* Использование средств автоматизации управления разработкой программного обеспечения.

Итогами прохождения второго модуля практики является выполнение всех поставленных целей и задач в процессе выполнения определенных видов работ, связанных с будущей профессиональной деятельностью.

# Список использованных источников

Стандарты

1. ГОСТ 7.1. – 2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2004. – 169 с.
2. ГОСТ 7.32 – 2001. Система стандартов по информацию, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: ИПК Издательство стандартов, 2001. – 21 с.
3. ГОСТ 7.82 – 2001. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2001. –21 с.
4. Единая система программной документации. – М.: Стандартинформ, 2005. –128 с.

Интернет-ресурсы

1. ПАО «НПП Сапфир» – [Электронный ресурс]. – Режим доступа: http://www.sapfir.ru/
2. Поразрядные операторы – [Электронный ресурс]. – Режим доступа: https://professorweb.ru/my/csharp/charp\_theory/level3/3\_12.php
3. Работа с файлами в С#. Классы StreamReader и StreamWriter – [Электронный ресурс]. – Режим доступа: http://mycsharp.ru/post/21/2013\_06\_12\_rabota\_s\_fajlami\_v\_si-sharp\_klassy\_streamreader\_i\_streamwriter.html

# Приложение А

Листинг программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

using System.Threading.Tasks;

namespace IntegratedCircuitEmulator

{

class ICEmulator

{

byte[] DIY = new byte[8], DIX = new byte[8], DI = new byte[16], INS = new byte[3];

char[] outputData;

int[] DO = new int[16];

bool CX, CY, EH, EL;

byte CRI8, CRI0;

bool[] V = new bool[2], G = new bool[2];

FileStream configFile, dataFile;

string[] pinsplace, data;

public void InputData()

{

if (InputFilePaths())

{

StreamReader reader = new StreamReader(configFile);

string[] configData = reader.ReadToEnd().Split('=', '\n');

pinsplace = new string[configData.Length / 2];

int j = 0;

for (int i = 2; i < configData.Length; i++)

{

if (configData[i].IndexOf("#") != -1)

configData[i] = configData[i].Substring(0, configData[i].IndexOf("#"));

if (i % 2 == 1)

{

pinsplace[j] = configData[i].Trim();

j++;

}

}

reader = new StreamReader(dataFile);

data = reader.ReadToEnd().Split(',');

data[0] = data[0].Substring(data[0].IndexOf('$'));

data[0] = data[0].Substring(data[0].IndexOf(' '));

for (int i = 0; i < data.Length - 1; i++)

data[i] = data[i].TrimStart();

reader.Close();

}

}

private void GetData(int i)

{

int k = 0;

outputData = new char[data[i].Length];

for (byte n = 0; n < data[i].Length; n++)

{

outputData[n] = data[i][n];

if (data[i][n] != ' ')

{

if (pinsplace[k].Contains("V"))

V[1 - Convert.ToByte(pinsplace[k][1].ToString())] = data[i][n] == '1';

if (pinsplace[k].Contains("G"))

G[1 - Convert.ToByte(pinsplace[k][1].ToString())] = data[i][n] == '1';

if (pinsplace[k].Contains("DIY"))

DIY[7 - Convert.ToByte(pinsplace[k][3].ToString())] = Convert.ToByte(data[i][n].ToString());

else

if (pinsplace[k].Contains("DIX"))

DIX[7 - Convert.ToByte(pinsplace[k][3].ToString())] = Convert.ToByte(data[i][n].ToString());

else

if (pinsplace[k].Contains("DI"))

{

byte ind = Convert.ToByte(pinsplace[k][2].ToString(), 16);

DI[15 - ind] = Convert.ToByte(data[i][n].ToString());

}

if (pinsplace[k].Contains("EH"))

EH = data[i][n] == '1';

if (pinsplace[k].Contains("EL"))

EL = data[i][n] == '1';

if (pinsplace[k].Contains("INS"))

INS[2 - Convert.ToByte(pinsplace[k][3].ToString())] = Convert.ToByte(data[i][n].ToString());

if (pinsplace[k].Contains("CRI8"))

CRI8 = Convert.ToByte(data[i][n].ToString());

if (pinsplace[k].Contains("CRI0"))

CRI0 = Convert.ToByte(data[i][n].ToString());

if (pinsplace[k].Contains("CX"))

CX = data[i][n] == '1';

if (pinsplace[k].Contains("CY"))

CY = data[i][n] == '1';

if (pinsplace[k].Contains("DO"))

{

byte ind = Convert.ToByte(pinsplace[k][2].ToString(), 16);

DO[ind] = n;

}

k++;

}

}

}

private bool InputFilePaths()

{

string path = "D5.pinsplace.txt";

if (File.Exists(path))

this.configFile = new FileStream(path, FileMode.Open, FileAccess.Read);

else

{

Console.WriteLine("Файл PINSPLACE по указанному пути не найден");

return false;

}

path = "ФК 1 D5 0000.txt";

if (File.Exists(path))

this.dataFile = new FileStream(path, FileMode.Open, FileAccess.ReadWrite);

else

{

Console.WriteLine("Файл с данными по указанному пути не найден");

return false;

}

return true;

}

public void Emulate()

{

InputData();

StreamWriter stream = new StreamWriter("OutputFile.txt", false, System.Text.Encoding.Default);

for (int i = 0; i < data.Length - 1; i++)

{

GetData(i);

string result = "";

if (CX && CY)

{

switch (String.Join("", INS))

{

case "000": result = CountLeadingZeroes(); break;

case "001": result = CodesMultiplication(); break;

case "101": result = Multiplication(); break;

case "010": result = LogicalLeftShift(); break;

case "110": result = ArithmeticLeftShift(); break;

case "011": result = LogicalRightShift(); break;

case "111": result = ArithmeticRightShift(); break;

}

result = result.PadLeft(16, '0');

OutputResult(i, result);

stream.Write(data[i]);

stream.WriteLine(",");

}

}

stream.Close();

}

private int ByteArrayToInt(byte[] s)

{

try

{

byte value;

int result = 0;

for (int i = 0; i < s.Length; i++)

{

value = s[i];

if (value == 1)

result += (int)Math.Pow(2, s.Length - 1 - i);

else if (value > 1)

throw new Exception("Invalid!");

}

return result;

}

catch

{

throw new Exception("Invalid!");

}

}

private void OutputResult(int ind, string result)

{

Console.WriteLine("Operation code " + String.Join("", INS) + " OutputData: " + result + " EH = " + EH.ToString() + " EL = " + EL.ToString());

for (int i = 0; i < 8; i++)

if (EH)

outputData[DO[i]] = result[i];

else

outputData[DO[i]] = 'L';

for (int i = 8; i < 16; i++)

if (EL)

outputData[DO[i]] = result[i];

else

outputData[DO[i]] = 'L';

data[ind] = String.Join("", outputData);

Console.WriteLine(data[ind]);

}

private string CountLeadingZeroes()

{

int R = 0, ZX = 0, ZY = 0, s = ByteArrayToInt(DI), y = ByteArrayToInt(DIY);

if (!Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4]))

{

for (int i = 0; i < DIX.Length; i++)

{

if (DIX[i] == 0) ZX++;

if (DIY[i] == 0) ZY++;

}

if ((ZX == 8) && (ZY == 8))

{

R = (1 << 11) + s + CRI8 + CRI0;

}

else

{

int I = ZX;

if (((1 << I + 1) >= y) && (y > (1 << I)))

{

R = (7 - I) \* (1 << 8) + Convert.ToInt32(String.Join("", DI).Substring(8)) + CRI0;

}

}

return Convert.ToString(R, 2);

}

return "LLLLLLLLLLLLLLLL";

}

private string Multiplication()

{

int R = 0, x = ByteArrayToInt(DIX), y = ByteArrayToInt(DIY), s = ByteArrayToInt(DI);

if (!V[0] && (G[0] || G[1]))

R = x \* y + s + CRI8 + CRI0;

if (V[0] && !V[1] && (G[0] || G[1]))

R = x \* y + (1 << 15) - 2 \* DIX[0] \* y - 128 + s + CRI8 + CRI0;

if (V[0] && V[1] && (G[0] || G[1]))

R = x \* y + (1 << 15) - 2 \* DIX[0] \* y + s + CRI8 + CRI0;

if (!(V[0] || V[1] || G[0] || G[1]))

R = x \* y + (1 << 15) - 2 \* Convert.ToByte(DIY[0]) \* x - 128 + s + CRI8 + CRI0;

if (V[0] && !V[1] && !(G[0] || G[1]))

R = x \* y + (1 << 16) + 4 \* DIX[0] \* Convert.ToByte(DIY[0]) - 2 \* DIX[0] \* y - 2 \* Convert.ToByte(DIY[0]) \* x - 256 + Convert.ToInt32(String.Join("", DI).Substring(1, 15)) + CRI8 + CRI0;

if (!V[0] && V[1] && !(G[0] || G[1]))

R = x \* y + (1 << 15) - 2 \* DIY[0] \* x + s + CRI8 + CRI0;

if (V[0] && V[1] && !(G[0] || G[1]))

R = x \* y + (1 << 16) + 4 \* DIX[0] \* DIY[0] - 2 \* DIX[0] \* y - 2 \* Convert.ToByte(DIY[0]) \* x - 128 + Convert.ToInt32(String.Join("", DI).Substring(1, 15)) + CRI8 + CRI0;

return Convert.ToString(R, 2);

}

private string CodesMultiplication()

{

int R, x = ByteArrayToInt(DIX), y = ByteArrayToInt(DIY), s = ByteArrayToInt(DI);

R = x \* y + s + CRI8 + CRI0;

return Convert.ToString(R, 2);

}

private string LogicalRightShift()

{

int R, x = ByteArrayToInt(DIX), y = ByteArrayToInt(DIY), s = ByteArrayToInt(DI), k = DIY[2] + DIY[1] + DIY[0];

if (((!G[0] && !G[1] && !Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4])) ||

(G[0] && !G[1] && Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4])) ||

(!G[0] && G[1] && !Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4])) ||

(G[0] && G[1] && Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4]))) && k != 0)

R = x \* (1 << (8 - k)) + CRI8 + CRI0 + s;

else if (!G[0] && !G[1] && !Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4]) && k == 0)

R = x \* (1 << 8) + CRI0 + Convert.ToInt32(String.Join("", DI).Substring(8));

else if (((!G[0] && !G[1] && Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4])) ||

(G[0] && !G[1] && !Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4])) ||

(!G[0] && G[1] && Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4]))) && k == 0)

R = x + s + CRI8 + CRI0;

else

R = s + CRI8 + CRI0;

return Convert.ToString(R, 2);

}

private string ArithmeticRightShift()

{

int R = 0, x = ByteArrayToInt(DIX), y = ByteArrayToInt(DIY), s = ByteArrayToInt(DI), k = DIY[2] + DIY[1] + DIY[0];

if (V[0] && V[1])

if (((G[0] && G[1] && Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4])) ||

(!G[0] && G[1] && !Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4])) ||

(G[0] && !G[1] && Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4]))) && k != 0)

R = x \* (1 << (8 - k)) + (1 << 15) - 2 \* DIX[0] \* (1 << (8 - k)) + CRI8 + CRI0 + s;

else if (!G[0] && !G[1] && !Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4]) && k != 0)

R = x \* (1 << (8 - k)) + (1 << 16) - 2 \* DIX[0] \* (1 << (8 - k)) + CRI8 + CRI0 + Convert.ToInt32(String.Join("", DI).Substring(1, 15));

if (V[0] && !V[1])

if (((G[0] && G[1] && Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4])) ||

(!G[0] && G[1] && !Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4])) ||

(G[0] && !G[1] && Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4]))) && k != 0)

R = x \* (1 << (8 - k)) + (1 << 15) - (1 << 7) - 2 \* DIX[0] \* (1 << (8 - k)) + CRI8 + CRI0 + s;

else if (!G[0] && !G[1] && !Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4]) && k != 0)

R = x \* (1 << (8 - k)) + (1 << 16) - (1 << 7) - 2 \* DIX[0] \* (1 << (8 - k)) + CRI8 + CRI0 + Convert.ToInt32(String.Join("", DI).Substring(1, 15));

if (!V[0])

if (((G[0] && G[1] && Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4])) ||

(!G[0] && G[1] && !Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4])) ||

(G[0] && !G[1] && Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4])) ||

(!G[0] && !G[1] && !Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4]))) && k != 0)

R = x \* (1 << (8 - k)) + CRI8 + CRI0 + s;

if (V[0] && V[1])

if (!G[0] && !G[1] && (Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4]) ||

!Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4]) ||

Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4])) && k != 0)

R = (1 << 16) + Convert.ToInt32(String.Join("", DI).Substring(1, 8)) + CRI8 + CRI0;

if (V[0] && !V[1])

if (!G[0] && !G[1] && (Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4]) ||

!Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4]) ||

Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4])) && k != 0)

R = (1 << 16) - (1 << 7) + Convert.ToInt32(String.Join("", DI).Substring(1, 15)) + CRI8 + CRI0;

if (!G[0] && !G[1] && !Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4]) && k == 0)

R = x + (1 << 8) + CRI0 + Convert.ToInt32(String.Join("", DI).Substring(8));

if (V[0] && V[1])

if (((!G[0] && !G[1] && Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4])) ||

(G[0] && !G[1] && !Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4])) ||

(!G[0] && G[1] && Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4]))) && k == 0)

R = x + (1 << 15) - 2 \* DIX[0] + CRI8 + CRI0 + s;

if (V[0] && !V[1])

if (((!G[0] && !G[1] && Convert.ToBoolean(DIY[3]) && !Convert.ToBoolean(DIY[4])) ||

(G[0] && !G[1] && !Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4])) ||

(!G[0] && G[1] && Convert.ToBoolean(DIY[3]) && Convert.ToBoolean(DIY[4]))) && k == 0)

R = x + (1 << 15) - 2 \* DIX[0] - (1 << 7) + CRI8 + CRI0 + s;

if (V[0] && V[1])

R = (1 << 15) + s + CRI8 + CRI0;

if (V[0] && !V[1])

R = (1 << 15) - (1 << 7) + s + CRI8 + CRI0;

if (!V[0])

R = s + CRI8 + CRI0;

return Convert.ToString(R, 2);

}

private string LogicalLeftShift()

{

int R, x = ByteArrayToInt(DIX), y = ByteArrayToInt(DIY), s = ByteArrayToInt(DI), k = DIY[2] + DIY[1] + DIY[0];

if ((G[0] && G[1] && !Convert.ToBoolean(DIY[4]) && !Convert.ToBoolean(DIY[3])) ||

(!G[0] && G[1] && !Convert.ToBoolean(DIY[4]) && Convert.ToBoolean(DIY[3])) ||

(G[0] && !G[1] && Convert.ToBoolean(DIY[4]) && !Convert.ToBoolean(DIY[3])) ||

(!G[0] && !G[1] && Convert.ToBoolean(DIY[4]) && Convert.ToBoolean(DIY[3])))

R = x \* (1 << k) + CRI8 + CRI0 + s;

else

R = s + CRI8 + CRI0;

return Convert.ToString(R, 2);

}

private string ArithmeticLeftShift()

{

int R = 0, x = ByteArrayToInt(DIX), y = ByteArrayToInt(DIY), s = ByteArrayToInt(DI), k = DIY[2] + DIY[1] + DIY[0];

if (V[0] && V[1])

if ((G[0] && G[1] && !Convert.ToBoolean(DIY[4]) && !Convert.ToBoolean(DIY[3])) ||

(!G[0] && G[1] && !Convert.ToBoolean(DIY[4]) && Convert.ToBoolean(DIY[3])) ||

(G[0] && !G[1] && Convert.ToBoolean(DIY[4]) && !Convert.ToBoolean(DIY[3])))

R = x \* (1 << k) + (1 << 15) - 2 \* x \* (1 << k) + CRI8 + CRI0 + s;

else if (!G[0] && !G[1] && Convert.ToBoolean(DIY[2]) && Convert.ToBoolean(DIY[3]))

R = x \* (1 << k) + (1 << 16) - 2 \* x \* (1 << k) + CRI8 + CRI0 + Convert.ToInt32(s.ToString().Substring(1, 15));

if (!V[0] && V[1])

if ((G[0] && G[1] && !Convert.ToBoolean(DIY[4]) && !Convert.ToBoolean(DIY[3])) ||

(!G[0] && G[1] && !Convert.ToBoolean(DIY[4]) && Convert.ToBoolean(DIY[3])) ||

(G[0] && !G[1] && Convert.ToBoolean(DIY[4]) && !Convert.ToBoolean(DIY[3])))

R = x \* (1 << k) + (1 << 15) - (1 << 7) - 2 \* x \* (1 << k) + CRI8 + CRI0 + s;

else if (!G[0] && !G[1] && Convert.ToBoolean(DIY[2]) && Convert.ToBoolean(DIY[3]))

R = x \* (1 << k) + (1 << 16) - (1 << 7) - 2 \* x \* (1 << k) + CRI8 + CRI0 + Convert.ToInt32(s.ToString().Substring(1, 15));

if (V[0] && !V[1])

if ((G[0] && G[1] && !Convert.ToBoolean(DIY[4]) && !Convert.ToBoolean(DIY[3])) ||

(G[0] && !G[1] && !Convert.ToBoolean(DIY[4]) && Convert.ToBoolean(DIY[3])) ||

(!G[0] && G[1] && Convert.ToBoolean(DIY[4]) && !Convert.ToBoolean(DIY[3])) ||

(!G[0] && !G[1] && Convert.ToBoolean(DIY[4]) && Convert.ToBoolean(DIY[3])))

R = x \* (1 << k) + CRI8 + CRI0 + s;

if (V[0] && V[1])

if ((!G[0] && !G[1] && !Convert.ToBoolean(DIY[4]) && !Convert.ToBoolean(DIY[3])) ||

(!G[0] && !G[1] && !Convert.ToBoolean(DIY[4]) && Convert.ToBoolean(DIY[3])))

R = (1 << 16) + CRI8 + CRI0 + Convert.ToInt32(s.ToString().Substring(1, 15));

if (!V[0] && V[1])

if (!G[0] && !G[1] && Convert.ToBoolean(DIY[4]) && !Convert.ToBoolean(DIY[3]))

R = (1 << 16) - (1 << 7) + CRI8 + CRI0 + s;

if (!V[0] && !V[1])

R = (1 << 15) + CRI8 + CRI0 + s;

if (V[0] && !V[1])

R = (1 << 15) - (1 << 7) + CRI8 + CRI0 + s;

if (V[0] && V[1])

R = (1 << 15) + CRI8 + CRI0;

return Convert.ToString(R, 2);

}

}

class Program

{

static void Main(string[] args)

{

ICEmulator IC = new ICEmulator();

IC.Emulate();

Console.ReadKey();

}

}

}