МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛ-ХОРАЗМИЙ

Факультет "Компьютерный инжиниринг"

Кафедра "Компьютерные системы"

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнение практических работ по предмету

«Основы компьютерной техники»

для бакалавров по направлению

5330700 - «Программное обеспечение информационных технологий»

Ташкент 2020

УДК 004.222.2

Авторы: PhD, старший преподаватель З.Ж. Алламуратова, доцент К.Т.Абдурашидова.

«Основы компьютерной техники» / ТУИТ, 35с. Ташкент, 2020.

Методические указания предназначены для закрепления знаний полученные по лекционным занятиям предмета "Основы компьютерной техники" и решениям задач по практическим работам данного предмета.

В методических указаниях приведены краткие рекомендации необходимые для решения задач, методика и примеры их выполнения, примеры задач для самостоятельных решений и подготовки практических заданий, по вопросам представления информации в ЭВМ в различных форматах и формах, выполнения арифметических и логических операций, методам минимизации логических функций, синтеза комбинационных схем, представления цифровых автоматов. Даны также задания для лучшего усвоения и закрепления теоретического и практического материала, рассмотренного на каждом занятии.

Методические указания составлены для студентов бакалавриатуры, обучающихся по направлении 5330700 «Программное обеспечение информационных технологий», разработана в соответствии с учебной программой дисциплины «Основы компьютерной техники» и предназначены в помощь студентам при подготовке к практическим занятиям по этой дисциплине.

Рецензенты:

А.Э.Мирзаев - PhD, доцент кафедры «Информационные технологии», ТУИТ.

Н.С.Маматов – д.т.н.ведущий научный сотрудник научно-инновационного центра Информационно-коммуникационных технологий при ТУИТ.

Введение

В настоящее время во всем мире компьютерная техника внедряется во все сферы человеческой деятельности и знания в этой области являются необходимыми для специалистов разрабатывающих программное обеспечение, в том числе и встроенное программного обеспечения. Таким образом, изучение дисциплины «Основы компьютерной техники» в настоящее время является актуальным для подготовки специалистов в области программного обеспечения информационных технологий.

Дисциплина «Основы компьютерной техники» читается в первом семестре и является базовой для всех технических дисциплин изучаемых студентами специальности 5330700 - «Программное обеспечение информационных технологий». По практическим занятиям предмета студенты закрепляют учебный материал по лекции, а так же:

- применение двоичной системы счисления, как для представления информации, так и для выполнения арифметико-логических операций;

- размещение в единой общей памяти команд и чисел фиксированной длины;

- линейную структуру адресации ячеек памяти, что требует наличия в процессоре счетчика команд;

- централизованное последовательное автоматическое считывание команд из памяти и интерпретацию их процессором.

Данная методическая рекомендация включает все практические работы по учебной программе дисциплины и содержит рекомендации по их выполнению, задачи для самостоятельной работы и варианты задач.

Результаты практической работы оформляются по требованию приведенного образца в данной рекомендации.

Практическое занятие №1. Переход из одной системы счисления в другую

Цель работы: Перевод чисел из десятичной системы счисления в двоичную, восьмеричную, шестнадцатеричную.

Теоретическая часть

Внутри компьютера информация всегда представляется в виде чисел, записанных в той или иной системе счисления.

Система счисления – совокупность приемов и правил для записи цифровыми знаками или символами.

При изучении материалов по системам счисления (раздел 1.1) необходимо уяснить, что десятичная система счисления является не единственно возможной системой, среди которых десятичная система имеет, пожалуй, единственное достоинство - она наиболее привычна для нас. Оперировать с другими системами счисления с равномерно распределенными весами не сложнее чем с десятичной системой, если учесть, что весовой коэффициент, связывающий два соседних разряда, определяется основанием системы, а не обязательно равен «10».

При изучении материалов по *переходу из одной системы счисления в другую* (*раздел 1.2*) необходимо уяснить, что самый удобный способ - это преобразование с использованием особого соотношения оснований, поэтому там, где это возможно, для преобразования нужно использовать именно его.

Преобразование двумя другими методами, приведенными в тексте лекций, выведены без задания конкретного значения оснований систем счисления, поэтому эти методы с одинаковым успехом могут быть использованы при переходах для любых системах счисления с *равномерно распределенными весами*, а не только в рассмотренных системах с основанием 10, 2, 16, 8. В этой связи будет полезным взять запись некоторого числа в системе счисления, отличной от перечисленных, (например с основанием «7») и перевести это число в другую систему (например в двоичную).

Следует обратить внимание на то, что преобразование дробных чисел необходимо предварять оценкой количества искомых разрядов преобразуемого числа в новом основании.

Переход из одной системы счисления в другую

Задача 1.

Перевести в десятичную систему двоичное число A = 110011001.

Решение

а) перевод методом с использованием весов разрядов:

$A = 110011001_2 = 256 + 128 + 16 + 8 + 1 = 409_{10}$

в) перевод методом деления:

11001 1001 | 1010

−1010 | 101000 частное 1 10100 0 | 1010

10 11 1010 | 100 последнее частное

− 1010 000 остаток 2

1001 остаток 1

Ответ: $A = 110011001_2 = 409_{10}$

Задача 2.

Перевести в десятичную систему двоичное число B = 0.1101.

Решение

а) перевод методом с использованием весов разрядов:

$A = 0.1100_2 = 0.50+0.25+0.06=0.81$ и после округления имеем:

$A = 0.1100_2 = 0.8_{10}$.

в) перевод методом умножения:

0. 1101

* 1010 новое основание

1.1010

+110.1000

1000.0010 первая смешанная дробь

* 1010

0.0100

Практическое занятие №1. Переход из одной системы счисления в другую

Цель работы: Перевод чисел из десятичной системы счисления в двоичную, восьмеричную, шестнадцатеричную.

Теоретическая часть

Внутри компьютера информация всегда представляется в виде чисел, записанных в той или иной системе счисления.

Система счисления – совокупность приемов и правил для записи цифровыми знаками или символами.

При изучении материалов по системам счисления (раздел 1.1) необходимо уяснить, что десятичная система счисления является не единственно возможной системой, среди которых десятичная система имеет, пожалуй, единственное достоинство - она наиболее привычна для нас. Оперировать с другими системами счисления с равномерно распределенными весами не сложнее, чем с десятичной системой, если учесть, что весовой коэффициент, связывающий два соседних разряда, определяется основанием системы, а не обязательно равен «10».

При изучении материалов по *переходу из одной системы счисления в другую* (*раздел 1.2)* необходимо уяснить, что самый удобный способ - это преобразование с использованием особого соотношения оснований, поэтому там, где это возможно, для преобразования нужно использовать именно его.

Преобразование двумя другими методами, приведенными в тексте лекций, выведены без задания конкретного значения оснований систем счисления, поэтому эти методы с одинаковым успехом могут быть использованы при переходах для любых системах счисления с *равномерно распределенными весами,* а не только в рассмотренных системах с основанием 10, 2, 16, 8. В этой связи будет полезным взять запись некоторого числа в системе счисления, отличной от перечисленных, (например с основанием «7») и перевести это число в другую систему (например в двоичную).

Следует обратить внимание на то, что преобразование дробных чисел необходимо предварять оценкой количества искомых разрядов преобразуемого числа в новом основании.

Переход из одной системы счисления в другую

Задача 1.

Перевести в десятичную систему двоичное число A = 110011001.

Решение

а) перевод методом с использованием весов разрядов:

$A = 110011001_2 = 256 + 128 + 16 + 8 + 1 = 409_{10}$

в) перевод методом деления:

11001 1001 | 1010

−1010 | 101000 частное 1 10100 0 | 1010

10 11 1010 | 100 последнее частное

− 1010 000 остаток 2

1001 остаток 1

Ответ: $A = 110011001_2 = 409_{10}$

Задача 2.

Перевести в десятичную систему двоичное число B = 0.1101.

Решение

а) перевод методом с использованием весов разрядов:

$A = 0.1100_2 = 0.50+0.25+0.06=0.81$ и после округления имеем:

$A = 0.1100_2 = 0.8_{10}$.

в) перевод методом умножения:

0. 1101

* 1010 новое основание

1.1010

+110.1000

1000.0010 первая смешанная дробь

* 1010

0.0100

+001.0000

00 1.0100 вторая смешанная дробь

Таким образом, имеем:

$A = 0.1100_2 = 0.81$ и после округления имеем:

$A = 0.1100_2 = 0.8_{10}$.

Задача 3.

Перевести в шестнадцатеричную систему двоичное число

$B = 11110000110.01101$.

Решение

Перевод можно выполнить с использованием особого отношения заданной и искомой систем счисления:

$B = 11110000110.01101_2 = 0111.1000.0110.0110.1000_2 = 7\ 8\ 6.\ 6\ 8_{16}$.

Задача 4.

Перевести в двоичную систему восьмеричное число $B = 174.1061_8$.

Решение

Перевод можно выполнить с использованием особого отношения заданной и искомой систем счисления:

$B = 174.1061_8 = 001.\ 111.100.001.000.110.001_2$.

Задача 5.

Перевести в двоичную систему число $B = 146_7$.

Решение

Перевод можно выполнить с использованием метода деления(умножения), при этом учитываем, что соседние разряды связывает соотношение q = 7:

```
146  |2     56 |2     26 |2     13 |2      5 |2     2 |2
-13  |56    -4 |26    -2 |13    -13 | 5    -4 |2    -2 1 -посл.частное
  16          16          6          0-остаток4  1-ост.5  0 –ост6
 -15         -15         -6
  1 -остаток1  1 -остаток2  0-остаток3
```

Ответ: $146_7 = 1010011_2$

Практическая часть

Задание:

1. Используя примеры решении задач выполните действии перехода из одной системы счисления в другую.

Для этой цели применяются следующие способы преобразований:

– метод преобразования с использованием весов разрядов в исходной и в искомой записи числа;

– метод деления (умножения) на новое основание;

– метод с использованием особого соотношения заданной и искомой систем счисления.

Контрольные вопросы:

1. Дайте определение позиционным и непозиционным системам счисления.
2. Чем характеризуется позиционная система счисления?
3. Приведите общую запись числа в системе с равномерно распределенными весами

Практическое занятие №2. Двоичная арифметика с положительными числами

Цель работы: Выполнения операции сложения и вычитания чисел, представленных в двоичной системе счисления, приобретение практических навыков выполнения операций над этими числами.

Теоретическая часть

Одна из основных целей, преследуемых при изучении арифметики с положительными числами - показать, что двоичная арифметика гораздо проще, чем привычная нам десятичная.

После изучения сложения и вычитания двоично-десятичных чисел необходимо уяснить, что в рассматриваемом случае выполнение этих

операций осуществляется по правилам двоичной арифметикой с выполнением коррекции при возникновении «переносе-заеме» через границы тетрад:

- по правилам двоичной арифметики заем из старшей тетрады приносит в тетраду, в которой был выработан сигнал о заёме, 16 , а должен принести по правилам работы с десятичными числами 10, поэтому, чтобы компенсировать лишнее, принесенное двоичным заеме, необходимо выполнить коррекция «-6» в соответствующей тетраде;
- по правилам двоичной арифметики перенос уносит из тетрада 16, а должен по правилам сложения десятичных чисел унести 10, поэтому тетрада, из которой выработался перенос корректируется на «+6»; кроме того «6» прибавляекся к тетрадам, в которых было получена недействительная десятичная цифра, т.е. значение превышающее 9 (двоичный код в тетраде 1001).

Задачи сложения и вычитания положительных двоичных чисел. Решение задач данного типа рассмотрено в примерах раздела 1.3.1, 1.3.2 текста лекций.

Умножение двоичных чисел (тип 1.3).

Для того чтобы выполнить умножение одного двоичного числа на другое, необходимо знать следующую простейшую таблицу умножения:

0 x 0 = 0

0 x 1 = 0

1 x 0 = 0

1 x 1 = 1

Кроме того, необходимо уметь складывать двоичные числа. Например, перемножение двоичных чисел 1001 и 1101 выполняется следующим образом:

1 0 01 – множимое

1 1 01 – множитель

1 0 01 – первое частичное произведение

+ 0 0 0 0 – второе частичное произведение

+ 1 0 01 – третье частичное произведение

10 01 – четвертое частичное произведение

11 10 101 - произведение

Задача 1.

Найти произведение 0.1011 и 0.1101, используя метод умножения «начиная со старшего разряда множителя со сдвигом промежуточного результата»

Решение

```
      0.1011
     *0.1101
       0000
      +1011
       1011
      1011
     +1011
    100001
   100001
    +0000
  1000010
 1000010
   + 1011
 10001111
```

Другие методы умножение двоичных чисел методами:

– начиная со старшего разряда со сдвигом множимого;

– начиная со младшего разряда со сдвигом множимого;

– начиная со старшего разряда со сдвигом множимого

рассмотрены в примерах 1- 3 раздела 1.3.3.

Деление положительных двоичных чисел.

Задача 1.

Найти C = A/B, если A = 1000110110, B = 11011.

Решение

1000110110

Ответ: C = 10100

Для самостоятельной подготовки предлагается решить соответствующие задачи из прилагаемого сборника задач.

Практическая часть

Задание:

- Найти разность двух десятичных чисел с использованием двоично-десятичной системе счисления;
- Найти сумму двух десятичных чисел с использованием двоично-десятичной системы счисления.

Контрольные вопросы:

1. При учете возникших переносов какая классификация используется для разрядов складываемых операндов?
2. Как реализуется операция вычитания в компьютере?
3. Начиная с какого разряда (со старшего или младшего) выполняется отработка множителя
4. Что сдвигается - множимое или промежуточное произведение?

Практическое занятие №3. Арифметика с алгебраическими числами

Цель работы: Выполнение операции с двоичными числами при использовании прямого, дополнительного, обратные коды и их модифицированные формы

Теоретическая часть

Любая информация (числа, команды, записи и т. п.) представляется в компьютере в виде двоичных кодов фиксированной или переменной длины. Для записи чисел также используют 32-разрядный формат (машинное слово), 16-разрядный формат (полуслово) и 64-разрядный формат (двойное слово). При работе с алгебраическими числами используются прямой, Дополнительный, обратный коды и их модифицированные формы.

При рассмотрении дополнительного и обратного кодов необходимо уяснить, что они используются при работе с алгебраическими числами. Эти коды очень похожи и каждый из них сопровождается операцией «+1» в младший разряд:

- в случае обратного кода операцией «+1» в младший разряд не используется при переходе из обратного кода в прямой и наоборот после инвертирования, но имеет место при переносе из знакового поля, который может возникнуть в процессе сложения;
- в случае дополнительного кода операцией «+1» в младший разряд имеет место при преобразовании из дополнительного кода в прямой и наоборот после инвертирования, но её нет при переносе из знакового поля в процессе сложения.

Сложение чисел в обратном коде

В обратном коде, как и в дополнительном операция вычитания заменяется операцией сложения. При этом знаковый разряд и цифровая часть рассматриваются так же как единое целое, вследствие чего машина оперирует с отрицательными числами как с неправильными дробями. Правильный знак суммы получается автоматически в процессе суммирования цифр знаковых разрядов операндов и единицы переноса из цифровой части, если она есть. Характерной особенностью обратного кода является наличие циклического переноса (если он возникает) из знакового разряда в младший разряд цифровой части, благодаря которому осуществляется коррекция суммы на 2^{-n}.

Считаем, что $|X_1| > |X_2|$ и $[X_1]_{об} + [X_2]_{об} < 1$

Пусть заданы два числа: $X_1 = 0{,}10110$; $X_2 = 0{,}00101$. Рассмотрим четыре случая:

1) $X_1 > 0$; $X_2 > 0$; $X_3 = X_1 + X_2 > 0$

Так как обратный код положительных чисел не отличается от самих чисел, то определение их суммы совпадает с ее получением в прямом коде:

$$[X_3]_{об} = [X_1]_{об} + [X_2]_{об} = X_1 + X_2$$

Правильный знак суммы получается автоматически, так как знаковые цифры равны 0, а $[X_1]_{об} + [X_2]_{об} < 1$ по условию, т.е. нет 1 переноса в знаковый разряд.

$X_1 = +0{,}10110$

$X_2 = \ 0{,}00101$

$X_3 = \ 0{,}11011$ Результат: $[X_3]_{об} = [X_3]_{пр} = 0{,}11011$

2) $X_1 > 0$; $X_2 < 0$; $X_3 = X_1 + X_2 > 0$

Сумма положительна, следовательно результат должен быть получен в прямом коде. Полученная сумма отличается от истинной на $(2\text{-}2^{-n})$, то есть нужна коррекция на эту величину. Коррекция на 2 получается автоматически (в разрядной сетке нет места для 1 переполнения знакового разряда), коррекция на 1 младшего разряда получается путем прибавления в младший разряд суммы единицы переполнения, которая возникает при сложении цифр знаковых разрядов, т.е. выполняется так называемый циклический перенос.

$[X_1]_{об} = +0{,}10110$

$[X_2]_{об} = 1{,}11010$

$10{,}10001$

$\longrightarrow \quad +1$

$[X_3]_{об} = 0{,}10001$ Результат: $[X_3]_{об} = 0{,}10001$

$[X_3]_{пр} = 0{,}10001$

3) $X_1 < 0$; $X_2 > 0$; $X_3 = X_1 + X_2 < 0$

Сумма отрицательна, значит, результат должен получиться в обратном коде, т.е. результат коррекции не требует.

$[X_1]_{об} = +1,01001$

$[X_2]_{об} = 0,00101$

$[X_3]_{об} = 1,01110$ Результат: $[X_3]_{об} = 1,01110;$

$[X_3]_{пр} = 1,10001$

4) $X_1 < 0$; $X_2 < 0$; $X_3 = X_1 + X_2 < 0$.

Сумма отрицательна, следовательно, результат должен быть получен в обратном коде. Как и во втором случае необходима коррекция результата только на единицу младшего разряда, так как для изображения 2 в разрядной сетке нет места. Поправка вносится аналогично второму случаю путем циклического переноса.

$[X_1]_{об} = +1,01001$

$[X_2]_{об} = 1,11010$

11,00011

$\longrightarrow$ +1

$[X_3]_{об} = 1,00100$ Результат: $[X_3]_{об} = 1,00100;$

$[X_3]_{пр} = 1,11011$

Задача 1

Найти двоично-десятичные значения C1, C2, C3, C4, определяемых выражениями:

C1= A+B, C2=A-B, C3= B-A, C4=-A-B,

используя модифицированный дополнительный код, если

A = 3 7 8 3, B = -5 49 2.

При реализации операции сложения использовать модифицированный дополнительный код.

Решение

Прямой код заданных двоично-десятичный чисел имеет вид:

$A]_{пк} = 0.\ 0011\ 0111\ 1000\ 0011$

$[B]_{пк} = 1.\ 0101\ 0100\ 1001\ 0010.$

Расчет выражений для C1, C2, C3, C4 осуществляется следующим образом.$[C1]_{пк}$:

	00.0011 0111 1000 0011	$[A]_{мок}$
+	11.1010 1011 0110 1110	$[B]_{пк} + 1 = [B]_{дк}+6$
	11.1110 0010 111 1 0001	- сумма ($[A]_{мок}$ и $[B]_{пк}0$), сформированная по правилам двоичного суммирования
+	0110 0110	- коррекция в тедрадах, где был перенос
	11.1110 1000 1111 0111	
	11. 0001 0111 0000 1000	- $[C1]_{мпк}$
	+1	
	11. 0001 0111 0000 1001	
	- 1 7 0 9	- $C1_{10}$ (десятичный эквивалент C1).

При выполнении первого суммирования по правилам двоичной арифметики возникающий перенос (в тетрадах, отмеченных знаком «*») унес лишнюю «6» (двоичный перенос унес из тетрады «16», а по правилам десятичного сложения он должен унести «10»), что означает, что избыточная «6», введенная за счет использования инверсного кода числа вместо обратного, исчезает в тех тетрадах, где был перенос, и сохраняется в тетрадах, где перенос отсутствовал. Коррекция на «+6» выполняется в тех тетрадах, где был перенос, чтобы ввести избыточную шестерку во все тетрады.

Таким образом после коррекции во всех тетрадах будет иметь место избыточная «6», что позволяет перейти от такой записи к прямому коду результата за счет инвертирования записей всех тетрад модульной части.

$[C4]_{пк}$:

	* *	
	11.1100 1000 0111 1101	$[-A]_{пк}+1 = [-A]_{дк}+6$
+	00.010 1 010 0 1001 0010	$[-B]_{мок}$
	100.0001 1101 0000 1111	
	00.0001 1101 0000 1111	перенос из знакового поля игнорируется

+ 1010 1010 - коррекция в тетрадах, где не было

переноса, с блокировкой переноса из тетрады

00. 0 0 0 1 0 1 1 1 0 0 0 0 1 0 0 1 $[C4]_{опк}=[C4]_{мпк}$

\+ 1 7 0 9 - $C4_{10}$ (десятичный эквивалент).

При выполнении первого суммирования по правилам двоичной арифметики возникающий перенос (в тетрадах, отмеченных знаком «*») унес лишнюю «6» (двоичный перенос унес из тетрады «16», а по правилам десятичного сложения он должен унести «10»), что означает, что избыточная «6», введенная за счет использования инверсного кода числа вместо обратного, исчезает в тех тетрадах, где был перенос, и сохраняется в тетрадах, где перенос отсутствовал. Коррекция на «10» выполняется в тех тетрадах, где не было переноса (прибавлением «10», которая является дополнительным кодом «-6», заменяется операция вычитания «6»).

Таким образом после коррекции во всех тетрадах будет иметь место точное значение, а так как результат положительный, то эта запись соответствует искомому прямому коду С1.

Практическая часть

Задание:

- Найти значения выражений в задачах вариантов используя модифицированный обратный код;

- Для правильных дробей эта точка предполагается расположенной между полем знака и полем модуля;

- Найти методом деления без восстановления остатка, используя обратный код, C = A/B, если

$[A]_{пк}$ = 0.1001;

$[B]_{пк}$ = 1.1110.

Контрольные вопросы:

1. Приведите правило формирования модуля обратного кода отрицательного двоичного числа

2. Обясните правило формирования модуля дополнительного кода отрицательного числа
3. Обясните процесс получения обратного кода.
4. Для чего используются дополнительный, прямой, обратный коды?

Практическое занятие №4. Арифметика с плавающей точкой

Цель работы: ознакомление с алгоритмами выполнения операции сложения и вычитания чисел, представленных в формате с плавающей точкой.

Теоретическая часть

В нормальной форме числа записываются в виде

$$A_н \doteq m_A q^p$$

где m_A – мантисса числа А;

q – основание системы счисления;

p – порядок числа А.

Для определенности вводятся ограничения $q^{-1} \le |m_A| \le 1$. Такая форма представления числа называется нормализованной.

Формат машинного изображения числа с плавающей запятой содержит разряд знака порядка, числовое поле порядка (7 бит), разряд знака мантиссы, числовое поле мантиссы, либо знак числа (0 разряд), характеристику, мантиссу. Кодирование знаков остается таким же, как было с фиксированной запятой.

А = - 110,1001. Преобразуем его в следующий вид А = - $0{,}1101001 * 2^{11(3)}$ и поместим в разрядной сетке. В 0-ом разряде – знак порядка 0 (+), с 1-го по 7-ой разряд запишем тройку в двоичном коде (11), поскольку запятую перенесли на три разряда влево, в 8-ом разряде знак мантиссы «1» (-), с 9-го по 15-ый разряд – мантисса числа – 1101001.

0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	1

Двоично-кодированные десятичные числа могут быть представлены в вычислительной машине полями переменной длины в так называемых упакованном и распакованном форматах

В упакованном формате для каждой десятичной цифры отводится по 4 двоичных разряда (полбайта), при этом знак числа кодируется в крайнем правом полубайте числа (1100 – знак «+» и 1101 – знак «-»).

Деление чисел с восстановлением остатка

При делении чисел с восстановлением остатка делитель вычитается из делимого и определяется знак нулевого (по порядку) остатка. Если остаток положительный, т.е. $|X_1| > |X_2|$, то в псевдознаковом разряде частного проставляется 1, при появлении которой формируется признак переполнения разрядной сетки и операция прекращается. Если остаток отрицательный, то в псевдознаковом разряде частного записывается 0, а затем производится восстановление делимого путем добавления к остатку делителя. Далее выполняется сдвиг восстановленного делимого на один разряд влево и повторное вычитание делителя. Знак получаемого таким образом остатка определяет первую значащую цифру частного: если остаток положителен, то в первом разряде частного записывается 1, если отрицательный , то записывается 0. Далее, если остаток положителен, то он сдвигается влево на 1 разряд и из него вычитается делитель для определения следующей цифры частного. Если остаток отрицателен, то к нему прибавляется делитель для восстановления предыдущего остатка, затем восстановленный остаток сдвигается на один разряд влево и от него вычитается делитель для определения следующей цифры частного и т.д. до получения необходимого количества цифр частного с учетом одного разряда для округления, т.е. до обеспечения требуемой точности деления. Вычитание делителя осуществляется сложением его псевдодополнительного кода.

Рассмотрим тот процесс на примере:

Необходимо разделить число + 0,1011 на число + 0,1101

На первом этапе определим знак произведения путем сложения по модулю 2 знаковых цифр сомножителей: 0 + 1 = 1, т.е. произведение будет иметь отрицательный знак.

Затем переходим непосредственно к умножению чисел:

0.1010 x 0.1101

0.00000000	0-ая сумма
+	
.1010	Множимое
.10100000 →	1-ая сумма
	1-ый сдвиг
.01010000	множимое
+	(блокируется)
.0000	
	2-ая сумма
.01010000 →	2-ой сдвиг
.00101000	Множимое
+	
.1010	
.11001000 →	3-я сумма
.01100100	3-ий сдвиг
+	
.1010	Множимое
1.00000100 →	4-ая сумма
.10000010	4-ий сдвиг

Результат произведения с учетом знака: 1.10000010

Ввиду того, что изображение положительного числа в дополнительном коде не отличается от его изображения в прямом коде, сумма двух чисел определяется по общим правилам:

$$[X_3]_д = [X_1]_д + [X_2]_д = X_1 + X_2$$

Практическая часть

Задания для закрепления:

Выполнить сложение нижеприведенных чисел в прямом, дополнительном и обратном коде:

1. $X_1 = -11011$; $X_2 = +11101$;
2. $X_1 = +10011$; $X_2 = -11001$;
3. $X_1 = +10101$; $X_2 = +11101$;

4. $X_1 = -110$; $X_2 = -111$;
5. $X_1 = +0,1011$; $X_2 = +0,1101$;
6. $X_1 = -0,1101$; $X_2 = -0,1111$

Контрольные вопросы:

1. Из каких частей состоит число с плавающей точкой?
2. Обясните нормальную форму числа с плавающей точкой
3. Какая форма записи числа с плавающей точкой называется нормализованной?
4. Обясните изменение мантиссы при сложении чисел?

Практическое занятие №5. Синтез логических схем

Цель работы: ознакомление с основными способами представления функций алгебры логики: табличным, матричным, геометрическим и аналитическим.

Теоретическая часть

Логическая (булевая) функция может быть представлена различными способами: матричным (табличным), геометрическим и аналитическим.

При матричном способе логическая функция задается таблицей истинности,, в левой части которой представлены все возможные двоичные наборы переменных данной функции, а в правой – указывается значение функции на этих наборах. Иногда двоичные наборы в таблице истинности удобно представлять также в виде матрицы, являющейся модификацией таблицы истинности. Например, задана некоторая логическая функция от трех переменных в виде таблицы истинности (таблица 1):

Таблица 1. Таблица истинности

№	Наборы $X_1X_2X_3$	Функция Y
0	0 0 0	0
1	0 0 1	1
2	0 1 0	1
3	0 1 1	1
4	1 0 0	0
5	1 0 1	1
6	1 1 0	1

7	1 1 1	0

Ее можно представить в виде сокращенной таблицы истинности, а также записать в следующем (числовом) виде:

Y = 1v2v3v5v6 = v: 1,2,3,5,6 = v (1,2,3,5,6);

Y = 0&4&7 = &: 0,4,7 = &(0,4,7)

Эту же функцию можно записать в виде матрицы, в узлах (клетках) которой указываются значения функции:

X_1 \ X_2X_3	00	01	10	11
0	0	1	1	1
1	0	1	1	0

X_3 \ X_1X_2	00	01	10	11
0	0	1	0	1
1	1	1	1	0

При геометрическом способе булевая функция f ($X_1X_2...X_n$) задается с помощью n-мерного куба (рис.1). Отмечая точками вершины куба, в которых функция принимает единичное значение, получим геометрическое представление функции.

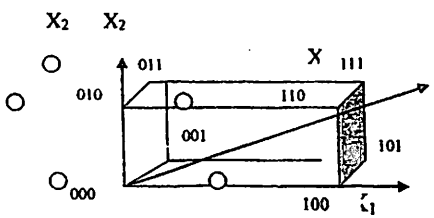

Рис.1. N – мерный куб

При аналитическом способе булевая функция задается формулами, т.е. аналитическими выражениями, построенными на основе операций булевой алгебры. Наиболее широкое распространение получила совершенная дизъюнктивная нормальная форма – СДНФ. Она записывается на основе таблицы истинности полностью определенной логической функции и

представляет собой дизъюнкцию минтермов данной функции. *Минтермом (конституентой единицы)* называется функция, принимающая значение 1 только на одном единственном наборе. Рассмотрим СДНФ функции от трех переменных, представленной выше таблицей истинности.

$$Y_{СДНФ}=\bar{X}_1\bar{X}_2X_3 \vee \bar{X}_1X_2\bar{X}_3 \vee \bar{X}_1X_2X_3 \vee X_1\bar{X}_2X_3 \vee X_1X_2\bar{X}_3$$

Применяя правила булевой алгебры можно перейти к более простым дизъюнктивным формам представления данной функции (ДНФ)

$$Y_{ДНФ} = \bar{X}_1X_3(\bar{X}_2 \vee X_2) \vee \bar{X}_2X_3(\bar{X}_1 \vee X_1) \vee \bar{X}_1X_2(\bar{X}_3 \vee X_3) \vee$$
$$X_2X_3(\bar{X}_1 \vee X_1) = \bar{X}_1X_3 \vee \bar{X}_2X_3 \vee \bar{X}_1X_2 \vee X_2X_3$$

Еще одной формой представления является совершенная конъюнктивная нормальная форма (СКНФ). Она представляется как конъюнкция макстермов (конституент нуля) данной булевой функции.

Для рассмотренного выше примера СКНФ определяется по нулевым наборам таблицы истинности функции. От этой канонической формы путем применения законов булевой алгебры можно перейти к более простым конъюнктивным формам представления логической функции (КНФ).

$$Y_{СКНФ}=(X_1 \vee X_2 \vee X_3)(\bar{X}_1 \vee X_2 \vee X_3)(\bar{X}_1 \vee \bar{X}_2 \vee \bar{X}_3)$$
$$Y_{КНФ}=(X_1 \vee X_2 \vee X_3)(\bar{X}_1 \vee X_2 \vee X_3)(\bar{X}_1 \vee \bar{X}_2 \vee \bar{X}_3)=(X_2 \vee X_3)X_1\bar{X}_1(\bar{X}_1 \vee \bar{X}_2 \vee \bar{X}_3)=$$
$$=(X_2 \vee X_3)(\bar{X}_1 \vee \bar{X}_2 \vee \bar{X}_3)$$

Практическая часть

Задание:

- Найти значения выражений в задачах вариантов используя модифицированный обратный код;
- Для правильных дробей эта точка предполагается расположенной между полем знака и полем модуля.

Контрольные вопросы:

1. Какие функции называются логическими?

2. Какие логические элементы имеются в базисе Буля?
3. Обясните из каких логических выражении составляются таблица истинности
4. Формы представления логической функции

Практическое занятие №6. Минимизация логических выражений

Цель работы: ознакомление с основными методами минимизации логических функций, получение практических навыков использования этих методов при минимизации функций алгебры логики.

Теоретическая часть

Минимизация (упрощение) логических функций состоит из следующих этапов:

1. Нахождение СДНФ (СКНФ) логической функции.
2. Переход от СДНФ к сокращенной ДНФ (сДНФ).
3. Переход от сокращенной ДНФ к тупиковой ДНФ(тДНФ)
4. Переход к заданному базису (если требуется).

На первом этапе осуществляется нахождение на основе таблицы истинности логической функции ее канонической формы (СДНФ или СКНФ).

Затем, используя различные законы и правила булевой алгебры, в частности, склеивание соседних импликант (имплициент), находится сокращенная форма ДНФ (КНФ).

На третьем этапе в сокращенной ДНФ (сКНФ) находят лишние импликанты (имплициенты) и убирают их. В результате получается тупиковая ДНФ (тКНФ), не имеющая лишних импликант (имплициент).

И, наконец, если есть в этом необходимость, осуществляется переход к тому или иному базису, используя законы де'Моргана.

Существует множество методов минимизации логических функций, Наибольшее распространение получили следующие:

1) аналитический (расчетный);

2) расчетно-табличный (метод Квайна);

3) табличный метод (метод Вейча-Карно).

Рассмотрим их подробнее.

Расчетный метод минимизации.

По таблице истинности запишем СДНФ заданной логической функции. Затем, применяя операцию склеивания $Ax \vee A\bar{x} = A(x \vee \bar{x}) = A$ к соседним минтермам, понизим на единицу ранг импликант. Если какой-либо минтерм не является соседним к остальным, то он записывается без изменений. Постепенно, понижая по возможности ранги импликант, получим сокращенную ДНФ, применительно к которой уже нельзя применить последующее склеивание. Затем переходим к этапу выявления лишних импликант.

Любая импликанта становится равной 1 лишь на одном определенном наборе значений истинности своих аргументов. Если на этом наборе сумма остальных импликант также обращается в единицу, то рассматриваемая импликанта является лишней.

Отбросив лишнюю импликанту, получим тупиковую. Если лишних импликант оказывается несколько, то следует отбросить одну и затем вновь проверить оставшиеся импликанты на наличие лишней.

Если исходной формой является СКНФ, то методика проверки на наличие лишней имплициенты остается той же, только имплициента приравнивается 0, и в случае лишней имплициенты оставшаяся часть функции равна 0.

Рассмотрим в качестве примера логическую функцию, приведенную на предыдущем занятии.

Таблица 2. Таблица истинности.

№	X_1	X_2	X_3	Y
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1

5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

$$Y_{СДНФ} = \overline{X_1}\,X_2\,\overline{X_3} \vee X_1\,\overline{X_2}\,\overline{X_3} \vee X_1\,X_2\,\overline{X_3} \vee X_1\,X_2\,X_3$$

$$Y = \overline{X_1}X_2\overline{X_3} \vee X_1\overline{X_2}\overline{X_3} \vee X_1X_2\overline{X_3} \vee X_1X_2X_3 = \overline{X_2}\,\overline{X_3}\left(\overline{X_1} \vee X_1\right) \vee X_1\overline{X_1}\left(\overline{X_2} \vee X_2\right) \vee X_1X_2\left(\overline{X_3} \vee X_3\right) =$$

$$= \overline{X_2}\,\overline{X_3} \vee X_1\overline{X_3} \vee X_1X_2$$

$$Y_{сДНФ} = \overline{X_2}\,\overline{X_3} \vee X_1\overline{X_3} \vee X_1X_2$$

Определим лишнюю импликанту:

1) $\overline{X_2}\,\overline{X_3} = 1 \to X_2 = 0 \quad X_3 = 0 \to Y' = X_1 * 1 \vee X_1 * 0 = X_1 \neq 1$ - не лишн;

2) $X_1\overline{X_3} = 1 \to X_1 = 1 \quad X_3 = 0 \to Y' = \overline{X_2} * 1 \vee X_2 * 1 = 1$ - лишняя;

3) $X_1X_2 = 1 \to X_1 = 1 \quad X_2 = 1 \to Y' = 0 * X_3 \vee 1 * \overline{X_3} = \overline{X_3} \neq 1$ - не лишн

Отбросим лишнюю импликанту $X_1\overline{X_3}$ и получим тупиковую форму функции: $Y_{тДНФ} = \overline{X_2}\,\overline{X_3} \vee X_1X_2$

Если в качестве исходной формы рассматривать СКНФ функции, то тогда

$$Y_{СКНФ} = \left(X_1 \vee X_2 \vee \overline{X_3}\right)\left(X_1 \vee X_3 \vee \overline{X_2}\right)\left(X_1 \vee \overline{X_2} \vee \overline{X_3}\right)\left(\overline{X_1} \vee X_2 \vee \overline{X_3}\right) =$$

$$= \left(X_1 \vee \overline{X_3}\right) * X_2 * \overline{X_2} * \left(X_2 \vee \overline{X_3}\right) * X_1 * \overline{X_1} * \left(X_1 \vee \overline{X_2}\right) * X_3 * \overline{X_3} =$$

$$= \left(X_1 \vee \overline{X_3}\right)\left(X_2 \vee \overline{X_3}\right)\left(X_1 \vee \overline{X_2}\right)$$

$$Y_{сКНФ} = \left(X_1 \vee \overline{X_3}\right)\left(X_2 \vee \overline{X_3}\right)\left(X_1 \vee \overline{X_2}\right)$$

1) $X_1 \vee \overline{X_3} = 0 \to X_1 = 0 \quad X_3 = 1 \to Y' = (X_2 \vee 0)(0 \vee \overline{X_2}) = 0$ -лишн

2) $X_2 \vee \overline{X_3} = 0 \to X_2 = 0 \quad X_3 = 1 \to Y' = (X_1 \vee 0)(X_1 \vee 1) = X_1 * 1 = X \neq 0$ не лишн

3) $X_1 \vee \overline{X_2} = 0 \to X_1 = 0 \quad X_2 = 1 \to Y' = (0 \vee \overline{X_3})(1 \vee \overline{X_3}) = \overline{X_3} * 1 = \overline{X_3} \neq 0$ не лишняя

$$Y_{тКНФ} = \left(X_2 \vee \overline{X_3}\right)\left(X_1 \vee \overline{X_2}\right)$$

Если необходимо перейти от тупиковой формы к тому или иному базису, можно воспользоваться теоремой де Моргана: $\overline{A \vee B} = \overline{A}\,\overline{B} \quad \overline{A * B} = \overline{A} \vee \overline{B}$

В нашем примере :

$$Y_{И\text{-}НЕ} = \overline{\overline{X_2}\,\overline{X_3} \vee X_1X_2} = \overline{\overline{X_2}\,\overline{X_3}}\;\overline{X_1X_2} \qquad Y_{ИЛИ\text{-}НЕ} = \overline{\left(X_2 \vee \overline{X_3}\right) \vee \left(X_1 \vee \overline{X_2}\right)}$$

Метод Квайна (расчетно-табличный метод)

В расчетно-табличном методе первый этап минимизации выполняется также, как и в расчетном, но для удобства и наглядности импликанты располагаются в колонках. В первой колонке записываются все минтермы (макстермы) СДНФ (СКНФ) логической функции и нумеруются. Затем они анализируются. Номера соседних минтермов записываются во вторую

колонку и рядом записывается общая импликанта. После сопоставления всех минтермов и выявления соседних, анализируется содержимое второй колонки. Если какой-то минтерм не участвовал в склеивании, то он записывается внизу второй колонки. И вновь все импликанты второй колонки нумеруются и процесс склеивания продолжается. Таким образом в первой колонке оказываются записанными все минтермы (*n*-разрядные импликанты), во второй колонке – импликанты *n-1* ранга и минтермы *n* - ранга, не принимавшие участия в склеивании, в третьей колонке – импликанты *n-2* ранга и т.д. до тех пор, пока дальнейшее склеивание не станет невозможным. В рассматриваемом примере:

1. $\overline{X_1}\,\overline{X_2}\,\overline{X_3}$ 1–2 $\overline{X_2}\,\overline{X_3}$
2. $X_1\overline{X_2}\,\overline{X_3}$ 2–3 $X_1\overline{X_3}$
3. $X_1X_2\overline{X_3}$ 3–4 X_2X_3
4. $X_1X_2X_3$

$$Y_{сДНФ} = \overline{X_2}\,\overline{X_3} \vee X_1\overline{X_3} \vee X_1X_2$$

Практическая часть

Задание:

- По заданной логической функции запишите таблицу истинности СДНФ;
- Используйте метод Карно для решения задачи по вариантам;
- Запишите таблицу истенности по СКНФ и минимизацию по Карно.

Контрольные вопросы:

1. Что такое минимизация и какие методы являются основными?
2. Как формировывается минимальное выражение логической функции?
3. Какие клетки в карте Карно являются логическими соседями?
4. Формы представления логической функции

Практическое занятие №7. Синтез цифровых автоматов

Цель работы: Синтез цифровых автоматов на примере автоматов Мили и Мура и построение таблиц переходов.

Теоретическая часть

Если рассматривать множество входных сигналов Z как множество букв входного алфавита, а множество выходных сигналов W - как множество букв выходного алфавита, то цифровой автомат можно рассматривать как преобразователь слов,т.е. входному слову, состоящему из S букв входного алфавита, он ставит в соответствие выходное слово, состоящее из S букв выходного алфавита. В этом случае эквивалентность двух цифровых автоматов можно определить следующим образом:

два цифровых автомата являются эквивалентными, если они, имея одинаковые множества входных и выходных сигналов, являются одинаковыми преобразователями слов, т. е. при любом начальном состоянии они для любого входного слова формируют одинаковые выходные слова.

Цифровой автомат называется конечным, если используемые для его задания множества конечны.

Цифровой автомат является полностью определенным, если каждой паре a(t), z(t) ставится в соответствие пара a(t+1), ϖ(t+1). В противном случае цифровой автомат называется частично определенным, или просто частичным.

Практически все накапливающие узлу ЭВМ можно рассматривать как цифровой автомат. Поэтому хорошо разработанная теория цифровых автоматов находит широкое применение для синтеза и анализа сложных схем в вычислительной техники.

Существует два основных вида цифровых автомата:

- автомат Мура;
- автомат Мили.

Автомат Мили определяется как автомат общего типа, т.е. его выходной сигнал и новое состояние являются функцией текущего состояния и действующего на входе сигнала.

Что же касается *автомата Мура*, то его выходной сигнал прямо не зависит от входного сигнала и определяется состоянием автомата, т.е. работа цифрового автомата типа Мура задается в виде уравнений:

- $a(t+1) = \delta\ (a(t), z(t))$;
- $\omega(t+1) = \lambda\ (a(t))$.

Зависимость выходного сигнала от входного в автомате Мура проявляется косвенно, т.е. через зависимость состояния от входного сигнала.

Задача. Синтезировать цифровой автомат, заданный в виде таблиц 1.

Память автомата построить на RS-триггере. При синтезе логических выражений использовать логический базис И, ИЛИ, НЕ.

Решение

Кодирование входных сигналов выполним через набор логических переменных «x». Множество входных сигналов включает три элемента. Поэтому для представления каждой из них достаточно использовать комбинации из двух переменных x_1, x_2. Кодировка входных переменных представлена таблицей 1.

Таблица1.

	ω_4 A_1	ω_2 A_2	ω_1 A_3	ω_3 A_4
z_1	A_3	A_3	A_4	–
z_2	A_3	–	A_2	A_1
z_3	A_2	A_1	A_1	A_3

Кодирование состояний выполним через набор логических переменных «Q». Множество состояний включает четыре элемента. Поэтому для представления каждой из них достаточно использовать комбинации из двух переменных Q_1, Q_2. Кодировка состояний представлена в таблице 2. Кодировка входных переменных представлена в таблице 3.

Табл.2

	x_1	x_2
z_1	0	1
z_2	1	0
z_3	1	1

Табл.3

	Q_1	Q_2
A_1	0	1
A_2	1	0
A_3	1	1
A_4	0	0

Для автомата Мура кодировка выходных сигналов не требуется, так как они однозначно связаны с состоянием, а следовательно и с кодами этих состояний.

Таблица 1 после замены переменных исходного задания цифрового автомата на их кодированные значения будут иметь вид, приведенный в таблице 4.

Таблица 4

	ω_3 $\overline{Q}_1 Q_2$	ω_2 $Q_1 \overline{Q}_2$	ω_1 $Q_1 Q_2$	ω_3 $\overline{Q}_1 \overline{Q}_2$
$\overline{x_1 x_2}$	11	11	00	—
$x_1 \overline{x_2}$	11	—	10	01
$x_1 x_2$	10	01	01	11

Таблица 5

	ω_3 $\overline{Q}_1 Q_2$	ω_2 $Q_1 \overline{Q}_2$	ω_1 $Q_1 Q_2$	ω_1 $\overline{Q}_1 \overline{Q}_2$
$\overline{x_1 x_2}$	11	11	00	—
$x_1 \overline{x_2}$	11	—	10	01
$x_1 x_2$	10	01	01	11

В таблице 4 клетки заполнены двухразрядным кодом, первый разряд которого отображает значение переменной Q_1, а вторая - переменную

Q_2(«1» - переменная имеет прямое значение, «0» - переменная имеет обратное значение).

Для управления памятью на RS- триггере, необходимо для первого и второго триггера, на которых строится память синтезируемого цифрового автомата, сформировать сигналы установки «1» (q_S) и установки «0» (q_R).

Логические выражения для этих сигналов, составленные на основе таблицы 4, имеют вид:

$$q_{S1} = \bar{Q}_1 Q_2\ \bar{x}_1 x_2 + \bar{Q}_1 Q_2\ x_1 \bar{x}_2 + \bar{Q}_1 Q_2\ x_1 x_2 + \bar{Q}_1 \bar{Q}_2\ x_1 x_2.$$

$$q_{R1} = Q_1 \bar{Q}_2\ x_1 x_2 + Q_1 Q_2\ \bar{x}_1 x_2 + Q_1 Q_2\ x_1 x_2 +$$

$$q_{S2} = Q_1 \bar{Q}_2\ \bar{x}_1 x_2 + Q_1 \bar{Q}_2\ x_1 x_2 + \bar{Q}_1 \bar{Q}_2\ x_1 \bar{x}_2 + \bar{Q}_1 \bar{Q}_2\ x_1 x_2.$$

$$Q_{R2} = \bar{Q}_1 Q_2\ x_1 x_2 + Q_1 Q_2\ \bar{x}_1 x_2 + Q_1 Q_2\ x_1 \bar{x}_2$$

Приведенные выражения формируются следующим образом:

– в дизъюнктивном выражении для q_{Si} используются конъюнкции, определяющие клетки таблицы, соответствующие случаям, когда в исходном коде состояния i-ый разряд имел значение «0», а конечном - «1»;

– в дизъюнктивном выражении для q_{Ri} используются конъюнкции, определяющие клетки таблицы, соответствующие случаям, когда в исходном коде состояния i-ый разряд имел значение «1», а конечном - «0».

Для более компактного представления полученных логических выражений и обозначений на формируемой схеме цифрового автомата, введем десятичную кодировку конъюнкций, используемых в полученных логических выражениях. Каждая конъюнкция представляет набор одних и тех же переменных Q_1, Q_2, x_1, x_2, поэтому представление конъюнкций можно рассматривать как четырехразрядный двоичный код и кодировать их десятичными эквивалентами этого двоичного кода. Таким образом, ранее полученные выражения можно представить в следующей компактной форме:

$q_{S1} = 5+6+7+3;$ $\quad q_{R1} = 11+13+15.$

$q_{S2} = 9+1+2+3;$ $\quad q_{R2} = 7+13+14.$

Логические выражения для выходных сигналов, исходя из таблицы 3 и 4, имеют вид:

$\omega_1 = Q_1 Q_2$; $\omega_2 = Q_1\bar{Q}_2$; $\omega_3 = Q_1\bar{Q}_2 + \bar{Q}_1\bar{Q}_2$

Сигнал ω_3 имеет место при двух состояниях цифрового автомата, поэтому соответствующее ему логическое выражение представляет собой дизъюнкцию двух конъюнкций, отражающих коды этих двух состояний автомата.

Практическая часть

Задание:

- Построить цифровой автомат заданного типа (Мили или Мура) для заданной ГСА, используя заданный тип триггера (RS-,D-, T- триггер).
- Построение устройства управления на базе цифрового автомата.

Контрольные вопросы:

1. Условия определенности цифрового автомата
2. Обясните зависимость выходного сигнала от входного?
3. Опишите принцип работы цифрового автомата Мура
4. Обясните переход от автомата Мили к автомату Мура

Практическое занятие №8. Составления микропрограммы

Цель работы: Составление микропрограммы для устройства управления по заданной граф - схемной алгоритме без использования модификатора дисциплины перехода.

Теоретическая часть

При микропрограммном принципе построения блока управления алгоритм выполнения операции осуществляется за счет выполнения особой программы, состоящей из отдельных команд, реализующих требуемую последовательность микроопераций. Такие команды получили название «микрокоманда», а совокупность микрокоманд - микропрограмма. Микропрограмма хранится в памяти.

При представлении алгоритма операции в виде ГСА выполняемая микрокоманды должна обеспечить выработку сигналов соответствующих

микроопераций и обеспечить переход к следующей микрокоманде, в том числе и при ветвлении вычислительного процесса. Таким образом, в формате микрокоманды в принципе необходимо иметь несколько полей:

- поле микроопераций (У), используемое для задания одной или нескольких микроопераций;
- поле условий (Х), в котором задаются проверяемые условия, влияющие на ветвление вычислительного процесса;
- поле адреса (А), в котором необходимо задавать информацию, определяющую следующую микрокоманду при возможном ветвлении (по крайней мере, по двум направлениям) для продолжения вычислительного процесса.

Задание всей перечисленной информации в едином формате микрокоманды затруднительно. Как правило, используют два вида, а, следовательно, и два формата микрокоманд:

- микрокоманды операционные;
- микрокоманды перехода.

Операционная микрокоманда используется для реализации операторных вершин ГСА. Её основная задача - задание выполняемой микрооперации. Формат операционной микрокоманды приведен на рис.4.4.1,а. Микрокоманда включает два поля:

- поле типа микрокоманды (Т);
- поле микрооперации (У).

Задача

Составить микропрограмму для реализации ГСА, приведенной на рис.8.1

Управляемый объект, характеризуется следующими параметрами:

- множество проверяемых условий

  $X = \{x_1, x_1, .. x_{25}.\}$;

- множество выполняемых микроопераций

  $Y = \{y_1, y_2, .. y_{120}, y_к\}$ ($y_к$- микрооперация , означающая последнюю микрокоманду микропрограммы);

– ёмкость памяти для записи микропрограмм

$V_{зу}$= 2кбайт = $2*2^{10}$ байт;

– длина ячейки памяти

L = 16 бит;

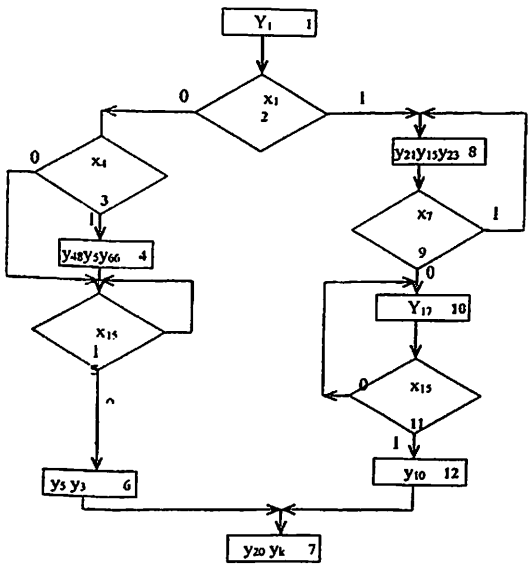

Рис. 8.1. Операторные вершины ГСА

– начальный адрес размещения составляемой микропрограммы в памяти

$A_н$=530.

Решение

Исходя из характеристик управляемого объекта, следует:

– длина поля для кодирования микроопераций равна к=7, так как количество выполняемых в объекте микроопераций равно 120 ($120 < 2^7$);

– длина поля для кодирования условий равна р=5, так как количество проверяемых условий в управляемом объекте равно 25 ($25 < 2^5$);

– длина кода адреса равна p=10, так как количеству адресов в памяти, учитывая, что длина адресуемой ячейки равна 16 бит, т.е. двум байтам, равно (1024 =<2^{10});.

Таким образом, формат микрокоманд для данного управляемого объекта имеет вид, приведенный на рисунке.

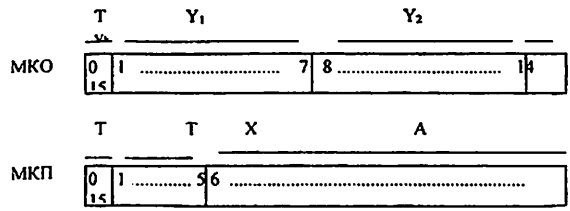

Рис. 8.2. Формат микрокоманд

Формат операционной микрокоманды (МКО) имеет длину 16 бит и включает:

– поле типа микрокоманды (М), имеющее длину в один бит и занимающее 0-ой разряд микрокоманды; в этом поле для данного типа микрокоманды записано значение «1»;

– поле первой микрооперации (Y_1), которое занимает разряды с 1-го по 7;

– поле второй микрооперации (Y_2), которое занимает разряды с 8-го по 14;

– поле микрооперации $y_к$, которое используется только в последней микрокоманде для указания завершения выполнения микропрограммы.

Формат микрокоманды перехода (МКП) имеет длину 16 бит и включает:

– поле типа микрокоманды (Т), имеющее длину в один бит и занимающее 0-ой разряд микрокоманды; в этом поле для данного типа микрокоманды записано значение «0»;

– поле проверяемого условия (Х), которое занимает разряды с 1-го по 5;

– поле адреса (А), которое занимает разряды с 6-го по 15.

Поле модификатора дисциплины переход в микрокоманде перехода отсутствует. Поэтому при составлении микропрограммы используется одна дисциплина перехода:

адрес следующей микрокоманды A_c формируется как:

$$A_c = \begin{cases} A_т + 1, \text{ если } x_i = 1; \\ A, \text{ если } x_i = 0, \end{cases}$$

где $A_т$. - адрес текущей выполняемой команды.

A - адрес перехода, располагаемый в одноименном поле выполняемой микрокоманды перехода.

При составления микропрограммы необходимо реализовать с помощью микрокоманд все действия вершины (включая и операционные и условные), имеющиеся в ГСА, и обеспечить имеющиеся ветвления процесса.

Микропрограмма, реализующая приведенную ГСА, имеет вид, приведенный в табл.1.

Таблица 1.

N пп	N вер.	Адрес расположения микрокоманды в ЗУ	Код микрокоманды	Примечание
	1.	2.	3.	4.
1.	1	1000010010 ($A_н$=530)	1. 0000001 . 0000000. 0	
2.	2	1000010011	0. 00001. 1000011100	3
3.	8	1000010100	1. 0010101. 0001111. 0	
4.	8'	1000010101	1. 0010111. 0000000. 0	
5.	9	1000010110	0. 00111. 1000011000	10
6.	-	1000010111	0. 00000 . 1000010100	8
7.	10	1000011000	1. 0010001 . 0000000. 0	
8.	11	1000011001	0. 01111. 1000011000	10
9.	12	1000011010	1. 0001010. 0000000. 0	
10.	7	1000011011	1. 0010100. 0000000. 1	

11.	3	1000011100	0. 00100. 1000011111	5
12.	4	1000011101	1. 0110000. 0000101. 0	
13.	4'	1000011110	1. 1000010. 0000000. 0	
14.	5	1000011111	0. 01111. 1000100001	6
15.	-	1000100000	0. 0000. 1000011111	5
16.	6	1000100001	1. 0000101. 0000011. 0	
17.	-	1000100010	0. 0000. 1000011011	7

В приведенной таблице:

- в первой, самой левой, колонке фиксируется номер строки;
- в первой графе (помечена «1») приводится номер вершины, реализуемой микрокомандой этой строки;
- во второй графе указан адрес расположения данной микрокоманды в запоминающем устройстве;
- в третьей графе располагается код микрокоманд;
- в четвертой графе указаны номера вершин (вершина-ссылка), адреса которых указываются в соответствующей команде перехода.

В приведенной микропрограмме кодировка микроопераций и проверяемых условий осуществлена по их индексам. Подчеркнутые коды адресов в микрокомандах перехода заполняют после записи последней строки формируемой микропрограммы, выбирая их из графы «Адрес» из строки, соответствующей номеру в графе «Примечание».

Микрокоманда в второй строке реализует первую вершину ГСА и поэтому записывается по адресу в ЗУ, соответствующему начальному адресу $A_н$, равному заданному начальному адресу 530 (в графе «Адрес» в второй строке записан двоичный эквивалент десятичного числа 530). Данная микрокоманда реализует операторную вершину, поэтому в поле «Т» кода микрокоманды имеет место «1». В реализуемой вершине задается одна микрооперация y_1, поэтому в поле Y1 записан двоичный семибитовый эквивалент её индекса «1», в поле Y2 записан двоичный

эквивалент «0», а в поле $y_к$ записан «0», так как данная микропрограмма реализует не последнюю вершину ГСА.

Микрокоманда в третьей строки реализует вторую вершину ГСА. Она в любом случае выполняется вслед за микрокомандой, реализующей вершину номер 1, поэтому записывается по адресу в ЗУ, на единицу большему, чем адрес расположения в ЗУ микрокоманды, реализующей вершину номер 1. Данная микрокоманда реализует условную вершину 2 заданного графа, поэтому в поле «Т» данной микрокоманды имеется значение «0». В поле «Х» данной микрокоманды записан пяти битовый двоичный эквивалент индекса проверяемого в реализуемой вершине (условие x_1). Поэтому в графе примечания записан номер этой вершины «3», а в следующем адресе располагается микрокоманда, реализующая вершину, расположенную по выходу «1» данной реализуемой вершины «2». Поле адреса «А» в данной микрокоманде и во всех других микрокомандах перехода первоначально не заполняется. Они заполняется после того, как будут записаны в память все микрокоманды формируемой микропрограммы.

При реализации вершины «8» необходимо задать три микрооперации, что нельзя сделать с помощью одной микрокоманды используемого формата. Поэтому данная вершина реализуется с помощью двух микрокоманд (строки «4» и «5»). Аналогичный случай имеет место при реализации вершины «4» (строки «13» и «14»).

После записи микрокоманды, реализующей вершину «6», необходимо расположить по следующему адресу в ЗУ микрокоманду, реализующую вершину «7». Однако вершина «7» уже представлена в микрокоманде (строка «10»). Поэтому в следующем адресе (строка «18») записывается команда безусловного перехода к микрокоманде, реализующей вершину «7». Команда безусловного перехода реализована на базе микрокоманды перехода при задании для проверки кода не существующего условия (в данном случае в качестве такого кода использован код «0000»).

Практическая часть

Задание:

- написать микропрограмму, соответствующую заданной ГСА, с учетом заданных множества микроопераций (Y), множества проверяемых условий (X), ёмкости запоминающего устройства (ЗУ) и начального адреса размещения микропрограммы (МП) в ЗУ;
- построение устройства управления с использованием микропрограммирования.

Контрольные вопросы:

1. Какие поля включает формат операционной микрокоманды в длину 16 бит?
2. Назовите форматы микрокоманды
3. Какие типа команд используется в микропрограммировании?
4. Классификация запоминающих устройств. Динамические запоминающие устройства.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Указ Президента Республики Узбекистан №УП-4947 от 7 февраля 2017 года «О стратегии действий по дальнейшему развитию Республики Узбекистан».

2. Столяров А.В. Программирование: введение в профессию. II:Низкоуровневое программирование. – М.МАКС Пресс, 2016.-496с.

3. Окулов С.М. Основы программирования [Электронный ресурс] / -8-е изд., перераб. – М.БИНОМ.Лаборатория знаний, 2015.

4. Гринченков Д.В. Математическая логика и теория алгоритмов для программистов: учебное пособие / Д.В.Гринченков, С.И. Потоцкий.- М.:КНОРУС, 2010. – 208 с.

5. Шень А. Программирование: теоремы и задачи. – 6-е изд., дополненное. – М.:МЦНМО, 2017. – 320 с.:ил.

6. Парфилова Н.И., Пруцков А.В., Пылькин А.Н., Трусов Б.Г. Информатика и программирование, основыинформатики. Учебник. 2012г.

7. Пупков А.Н., Самарин В.В., Царёв Р.Ю. Информатика и программирование. 2012 г.

8. Микушин А.В. Цифровые устройства и микропроцессоры / А.В.Микушин.- М.: БХВ-Петербург, 2010. -338 с.

9. Молоков К.А. "Основы информатики и программирование под Windows. Учебное пособие" – 2015 г. Изд.: Проспект-176 с.

10.Окулов С.М., Пестова А.А., Пестов О.А. Информатика в задачах. Издательство: Лаборатория базовых знаний, Бином. 2013 г.

СОДЕРЖАНИЕ

Методические указания по предмету «Основы компьютерной техники» для бакалавров по направлению 5330700 - «Программное обеспечение информационных технологий»

Рассмотрено на заседании кафедры «Компьютерные системы»
от 12.05.2020 г. Протокол № 22
Рассмотрено на заседании факультета «Компьютерный инжиниринг»
от 19.05.2020 г. Протокол № 20

Рассмотрено и рекомендовано к изданию на заседании
научно–методического Совета ТУИТ
от « 23 » 06 2020 г. Протокол № 9 (134)

Составители З.Ж. Алламуратова
К.Т.Абдурашидова

Рецензенты: А.Э.Мирзаев
Н.С.Маматов

Ответственный редактор: Ж.Х.Джуманов

Корректор: С.Х.Абдуллаева

Формат 60x84 1/16. Печ.лист 2,5.
Заказ № 84. Тираж 30.
Отпечатано в «Редакционно издательском»
отделе при ТУИТ.
Ташкент ул. Амир Темур, 108.