МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Р. Е. АЛЕКСЕЕВА»

Кафедра «Вычислительные системы и технологии»

ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ЭВМ И ОПЕРАЦИИ В ДВОИЧНЫХ КОДАХ

Методические указания к лабораторным работам по курсу «Организация ЭВМ» для студентов высших учебных заведений по направлению 09.03.01 «Информатика и вычислительная техника» профилей «Вычислительные машины, комплексы, системы и сети» и «Программное обеспечение средств вычислительной техники и автоматизированных систем» всех форм обучения

Нижний Новгород 2020

Составители: **П.С. Кулясов**
УДК 004.9

Представление информации в ЭВМ и операции в двоичных кодах: метод. указания к лаб. работам по курсу «Организация ЭВМ» для студентов высших учебных заведений по направлению 09.03.01 «Информатика и вычислительная техника» профилей «Вычислительные машины, комплексы, системы и сети» и «Программное обеспечение средств вычислительной техники и автоматизированных систем» всех форм обучения / НГТУ им. Р.Е. Алексеева; сост.: П.С. Кулясов – Нижний Новгород, 2020. – 15 с.

В методических указаниях описываются системы счисления и коды, применяемые для представления информации в ЭВМ. Приводятся подробные алгоритмы методов машинного умножения и их реализация на ЭВМ с помощью модели арифметико-логического устройства с программируемым устройством управления. Материал предназначен для студентов высших учебных заведений по направлению 09.03.01 «Информатика и вычислительная техника» профилей «Вычислительные машины, комплексы, системы и сети» и «Программное обеспечение средств вычислительной техники и автоматизированных систем».

Редактор

Подп. к печ. 26.10.2020. Формат . Печать .
Бумага . Усл. печ. л. 1. Тираж . Заказ.

Нижегородский государственный технический университет им. Р.Е. Алексеева.
Типография НГТУ, 603950, Нижний Новгород, ул. Минина, 24.

Содержание

1. Системы счисления и представление информации в ЭВМ

Всякое число представляется набором цифр. Способ представления чисел цифрами характеризует систему счисления (код). Системой счисления называется совокупность цифр и правил для записи чисел. На рис. 1 представлена классификация и краткая характеристика систем счисления.

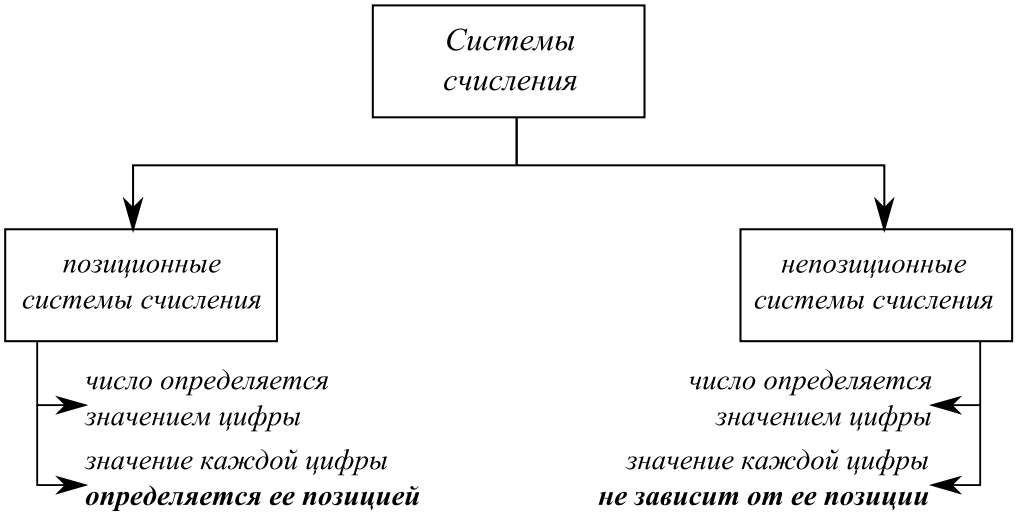

Рис. 1. Классификация и краткая характеристика систем счисления

Примером непозиционной системы счисления может служить римская система. Несмотря на то, что она до сих пор используется в некоторых узкоспециализированных областях, она крайне неудобна для выполнения любого рода вычислений.

Наибольшее распространение получили позиционные системы счисления, в которых число, эквивалентное записанной цифре, определяется как значением этой цифры, так и ее положением (позицией) среди других цифр в числе. Для записи чисел в позиционной системе счисления используется определенное количество графических знаков (цифр и букв), которые отличаются один от другого – основание системы счисления. В системах счисления с основанием меньше 10 для записи чисел используют только цифры, для систем с основанием больше 10 – цифры и буквы латинского алфавита (A, B, C, D и так далее). Отдельная позиция в записи числа называется разрядом, а номер позиции – номером разряда. Десятичная (дециальная) система счисления, используемая человеком в повседневной жизни – типичный пример позиционной системы.

Положительное число из i разрядов в позиционной системе с основанием a может быть представлено как $N_a = x_{i-1}a^{i-1} + x_{i-2}a^{i-2} + ... + x_1a^1 + x_0a^0$, где x – любая цифра от 0 до a-1. Здесь

первое слагаемое представляет собой старший разряд числа, а последнее – младший.

В десятичной системе, например, число 573 можно представить как $573_{10} = 5 \cdot 10^2 + 7 \cdot 10^1 + 3 \cdot 10^0$.

Система счисления должна обеспечивать выполнение следующих требований:

- возможность представления любого числа;
- простота выполнения арифметических операций;
- достижение высокого быстродействия машины в процессе обработки информации.

Двоичная система счисления, обладает следующими основными достоинствами:

- простота выполнения арифметических операций;
- наличие надежных микроэлектронных схем с двумя устойчивыми состояниями, предназначенных для хранения значений двоичного кода – 0 и 1.

В связи с этим двоичная система счисления является основной системой счисления, используемой в ЭВМ. Двоичные числа в компьютерах размещаются в ячейках памяти (регистрах), которые состоят из запоминающих элементов – триггеров. В одной ячейке хранится значение одного двоичного разряда – бит информации. Совокупность запоминающих элементов для размещения одного двоичного числа называется разрядной сеткой компьютера, а способ размещения компонентов числа в разрядной сетке – форматом. В компьютерах используют две формы представления числа:

- с фиксированной запятой – перед старшим разрядом числа для правильной дроби или после младшего разряда для целого числа.
- с плавающей запятой, место положения которой задается порядком числа.

В числах со знаком выделяют дополнительный знаковый разряд. В нем для положительного числа записывают ноль, а для отрицательного – единицу.

В цифровой технике применяют двоичные (бинарные) коды. Существует ряд двоичных кодов, каждый из которых обладает определенными свойствами. В цифровой технике наибольшее применение получил так называемый прямой двоичный код, в котором *i*-разрядное число представляется как: $N_2 = x_{i-1} \cdot 2^{i-1} + x_{i-2} \cdot 2^{i-2} + ... + x_1 \cdot 2^1 + x_0 \cdot 2^0$. Здесь x может иметь два значения – ноль и единица.

Порядок счета в прямом двоичном коде совпадает с порядком счета внутри каждого десятичного разряда, что упрощает взаимный перевод чисел десятичного и двоичного кодов. Этот двоичный код называют еще

кодом 8421 – по весовым коэффициентам (или, как еще говорят – весам) первых четырех разрядов числа.

В табл. 1 приведены десятичные числа от 0 до 15 и их эквиваленты в коде 8421.

Таблица 1

Код		**Код**		**Код**	
десятичный	**8421**	**десятичный**	**8421**	**десятичный**	**8421**
0	0000	6	0110	11	1011
1	0001	7	0111	12	1100
2	0010	8	1000	13	1101
3	0011	9	1001	14	1110
4	0100	10	1010	15	1111
5	0101				

Из таблицы следует, что для представления десятичных цифр от 0 до 9 (то есть одного десятичного разряда) требуется четыре двоичные цифры.

Кроме прямого применяются и другие коды, с помощью которых упрощаются арифметические действия. К ним относятся, в частности, обратный и дополнительный коды.

Для положительного числа его запись в прямом и обратном коде совпадают, для отрицательного обратный код получается путем инвертирования числовых разрядов числа (знаковый разряд не меняется). Дополнительный код числа образуется из обратного кода добавлением единицы к младшему разряду.

Широко применяется двоично-десятичный код, в котором цифры каждого разряда десятичного числа представляются четырехразрядным двоичным числом (тетрадой). Так число $N_{10} = 573$ в двоично-десятичном коде имеет вид $N_{2-10} = 010101110011$. Основное достоинство двоично-десятичного кода в простоте взаимного перевода двоичных и десятичных чисел.

2. Операции в двоичных кодах в ЭВМ. Методы машинного умножения

Для простоты описания алгоритмов два числа, которые необходимо перемножить принято обозначать как множимое (то число, которое умножается) и множитель (то число, на которое происходит умножение). Реализация умножения в машине осуществляется через выполнение операции сложения в цикле, причем число итераций цикла равняется количеству разрядов множителя. Все существующие методы машинного умножения сводятся к одной и той же последовательности действий:

Шаг 1. Анализ числового разряда множителя.

Шаг 2. Суммирование множимого с частичным произведением (в случае, когда анализируемый разряд множителя равен единице).

Шаг 3. Сдвиг множимого либо частичного произведения (для увеличения либо уменьшения, в зависимости от метода, его разрядности) и множителя (чтобы избежать повторного анализа его разрядов).

Различие между методами заключается в двух аспектах:

1. Анализируемый разряд множителя (старший либо младший).
2. Осуществление сдвига множимого либо частичного произведения.

Комбинации возможных вариантов этих двух аспектов и дают набор методов машинное умножение, количество которых равняется $2^2 = 4$:

1. Умножение с младших разрядов множителя с подвижным множимым.
2. Умножение с младших разрядов множителя с неподвижным множимым.
3. Умножение со старших разрядов множителя с подвижным множимым.
4. Умножение со старших разрядов множителя с неподвижным множимым.

Арифметические операции над числами в двоичных кодах в ЭВМ выполняются в специальном блоке процессора, называемом арифметико-логическое устройство (АЛУ).

3. Алгоритмы методов машинного умножения

3.1 Умножение с младших разрядов множителя с подвижным множимым

Шаг 1. Заносим множитель в регистр 1 в младшие разряды.

Шаг 2. Заносим знак множителя в сумматор.

Шаг 3. Заносим множимое в регистр 2 в младшие разряды.

Шаг 4. Заносим знак множимого в сумматор.

Шаг 5. Заносим в счетчик значение, равное количеству итераций цикла (равно количеству разрядов множителя).

Шаг 6. Анализируем младший разряд множителя. Если там единица – заносим множимое в сумматор, формируя частичное произведение. Если ноль – переход на шаг 7.

Шаг 7. Сдвигаем множитель на один разряд вправо, множимое – на один разряд влево.

Шаг 8. Уменьшаем значение счетчика на единицу.

Шаг 9. Сравниваем значение счетчика с нулем. Если не ноль – переход на шаг 6. Если ноль – конец работы алгоритма.

3.2 Умножение с младших разрядов множителя с неподвижным множимым

Шаг 1. Заносим множитель в регистр 1 в младшие разряды.
Шаг 2. Заносим знак множителя в сумматор.
Шаг 3. Заносим множимое в регистр 2 таким образом, что номер разряда регистра, в который заносится младший разряд множимого на единицу больше количества разрядов множителя (например, если количество разрядов множителя равно четырем, то множимое заносится в регистр, начиная с пятого разряда).
Шаг 4. Заносим знак множимого в сумматор.
Шаг 5. Заносим в счетчик значение, равное количеству итераций цикла (равно количеству разрядов множителя).
Шаг 6. Анализируем младший разряд множителя. Если там единица – заносим множимое в сумматор, формируя частичное произведение. Если ноль – переход на шаг 7.
Шаг 7. Сдвигаем множитель на один разряд вправо, содержимое сумматора – на один разряд вправо.
Шаг 8. Уменьшаем значение счетчика на единицу.
Шаг 9. Сравниваем значение счетчика с нулем. Если не ноль – переход на шаг 6. Если ноль – конец работы алгоритма.

3.3 Умножение со старших разрядов множителя с подвижным множимым

Шаг 1. Заносим множитель в регистр 2 в старшие разряды.
Шаг 2. Заносим знак множителя в сумматор.
Шаг 3. Заносим множимое в регистр 1 таким образом, что номер разряда регистра, в который заносится младший разряд множимого был равен количеству разрядов множителя (например, если количество разрядов множителя равно четырем, то множимое заносится в регистр, начиная с четвертого разряда).
Шаг 4. Заносим знак множимого в сумматор.
Шаг 5. Заносим в счетчик значение, равное количеству итераций цикла (равно количеству разрядов множителя).
Шаг 6. Анализируем старший разряд множителя. Если там единица – заносим множимое в сумматор, формируя частичное произведение. Если ноль – переход на шаг 7.
Шаг 7. Сдвигаем множитель на один разряд влево, множимое – на один разряд вправо.
Шаг 8. Уменьшаем значение счетчика на единицу.

Шаг 9. Сравниваем значение счетчика с нулем. Если не ноль – переход на шаг 6. Если ноль – конец работы алгоритма.

3.4 Умножение со старших разрядов множителя с неподвижным множимым

Шаг 1. Заносим множитель в регистр 2 в старшие разряды.
Шаг 2. Заносим знак множителя в сумматор.
Шаг 3. Заносим множимое в регистр 1 в младшие разряды.
Шаг 4. Заносим знак множимого в сумматор.
Шаг 5. Заносим в счетчик значение, равное количеству итераций цикла (равно количеству разрядов множителя).
Шаг 6. Анализируем старший разряд множителя. Если там единица – заносим множимое в сумматор, формируя частичное произведение. Если ноль – переход на шаг 7.
Шаг 7. Сдвигаем множитель на один разряд влево, содержимое сумматора – на один разряд влево.
Шаг 8. Уменьшаем значение счетчика на единицу.
Шаг 9. Сравниваем значение счетчика с нулем. Если не ноль – переход на шаг 6. Если ноль – конец работы алгоритма.

Следует отметить особенность данного метода машинного умножения, заключающуюся в том, что при его реализации количество сдвигов для содержимого сумматора должно быть на единицу меньшим, чем количество итераций в цикле (например, если количество разрядов множителя равно четырем и, следовательно, количество итераций цикла равно тоже четырем, то количество сдвигов сумматора должно быть равно трем). В связи с этой особенностью в алгоритм, при реализации данного метода, должны быть внесены соответствующие изменения.

4. Реализация методов машинного умножения

4.1 Модель арифметико-логического устройства

В рамках лабораторных работ по дисциплине «Организация ЭВМ» для реализации методов машинного умножения используется программная модель арифметико-логического устройства (АЛУ) с программируемым устройством управления (УУ), которая является симулятором, точно воспроизводящим конструктивные особенности реального устройства (разработка кафедры «Вычислительные системы и технологии», свидетельство о государственной регистрации программы для ЭВМ №2018616160 от 24.05.2018 г.).

Модель АЛУ (рис. 2) включает в себя два регистра (обозначены RGA и RGB), сумматор обратных кодов и память, которая используется для занесения данных в регистры (для удобства работы и упрощения проверки результатов выполнения операций они снабжены специальным полем, куда выводится их содержимое, переведенное в десятичную систему счисления). Память и регистры содержат один знаковый (ТЗН) и девять числовых разрядов, сумматор содержит два знаковых (ТЗН, который содержит знак числа, и ТПП, который используется в качестве индикатора переполнения) и девять числовых разрядов. Регистр RGA поддерживает анализ младшего числового разряда и имеет возможность сдвига содержимого числовых разрядов вправо, регистр RGB – анализ старшего числового разряда и сдвиг содержимого числовых разрядов влево.

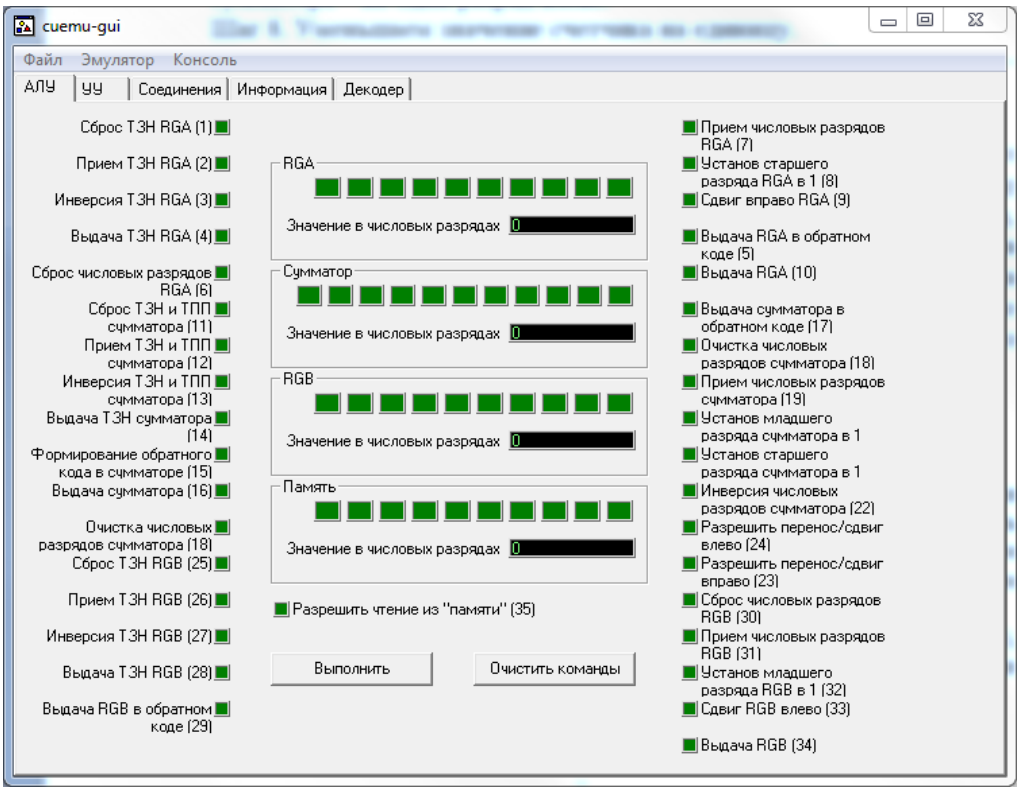

Рис. 2. Модель арифметико-логического устройства

Для выполнения операций в двоичных кодах модель АЛУ поддерживает 35 различных микроопераций (далее МО), каждая из которых снабжена соответствующим порядковым номером (табл. 2). Микрооперации в программе записаны в два столбца: в левом приведены микрооперации, позволяющие работать со знаковыми разрядами, в правом – с числовыми. Каждая микрооперация для ее выполнения должна быть включена (нажатием левой кнопки мыши на соответствующую кнопку) и снабжена индикатором (если он светится – микрооперация включена).

Таблица 2

Номер МО	Название МО	Выполняемое действие
1	Сброс ТЗН RGA	Сброс (установка в ноль) содержимого триггера знака (знакового разряда) регистра RGA
2	Прием ТЗН RGA	Прием из памяти либо сумматора содержимого знакового разряда в знаковый разряд регистра RGA
3	Инверсия ТЗН RGA	Инверсия (смена значения на противоположное) содержимого знакового разряда регистра RGA
4	Выдача ТЗН RGA	Выгрузка содержимого знакового разряда регистра RGA (для передачи в сумматор)
5	Выдача RGA в обратном коде	Выгрузка содержимого регистра RGA, преобразованного в обратный код (для передачи, например, в сумматор). Данная микрооперация необходима для выполнения операций над отрицательными числами, поскольку сумматор выполняет вычисления в обратных кодах, а для отрицательных чисел обратный код отличается от прямого. При выполнении операций над положительными числами эту микрооперацию можно не использовать, поскольку для них обратный код совпадает с прямым
6	Сброс числовых разрядов RGA	Сброс (установка в ноль) содержимого числовых разрядов регистра RGA
7	Прием числовых разрядов RGA	Прием значений в числовые разряды регистра RGA
8	Установ старшего разряда RGA в 1	Установка содержимого старшего числового разряда регистра RGA в единицу
9	Сдвиг вправо RGA	Сдвиг содержимого числовых разрядов регистра RGA вправо на один разряд
10	Выдача RGA	Выгрузка содержимого числовых разрядов регистра RGA (для передачи, например, в сумматор)
11	Сброс ТЗН и ТПП сумматора	Сброс (установка в ноль) содержимого триггера знака (знакового разряда) и триггера переполнения сумматора
12	Прием ТЗН и ТПП сумматора	Прием значений в знаковые разряды сумматора
13	Инверсия ТЗН и ТПП сумматора	Инверсия (смена значения на противоположное) содержимого знаковых разрядов сумматора
14	Выдача ТЗН сумматора	Выгрузка содержимого знакового разряда сумматора (ТЗН, содержащего знак числа, находящегося в сумматоре)
15	Формирование обратного кода в сумматоре	Микрооперация используется при осуществлении операций над отрицательными числами. Отрицательные числа выгружаются из регистров в сумматор в обратных кодах, сумматор осуществляет их сложение, после чего необходимо перевести результат из обратного кода в прямой, для этого используется данная микрооперация
16	Выдача сумматора	Выгрузка содержимого числовых разрядов сумматора (для приема в регистры)
17	Выдача сумматора в обратном коде	Выгрузка содержимого числовых разрядов сумматора, переведенного в обратный код
18	Очистка числовых разрядов сумматора	Сброс (установка в ноль) содержимого числовых разрядов сумматора
19	Прием числовых	Прием значений в числовые разряды сумматора (из

	разрядов сумматора	регистров, выгрузка из памяти в сумматор напрямую, минуя регистры, не поддерживается)
20	Установ младшего разряда сумматора в 1	Установка содержимого младшего числового разряда сумматора в единицу
21	Установ старшего разряда сумматора в 1	Установка содержимого старшего числового разряда сумматора в единицу
22	Инверсия числовых разрядов сумматора	Инверсия (смена значения на противоположное) содержимого числовых разрядов сумматора
23	Разрешить перенос/сдвиг вправо	Данная микрооперация имеет два различных назначения. Первое – разрешение переноса вправо (в младший числовой разряд) при выполнении сумматором операции сложения. Второе – сдвиг вправо на один разряд содержимого числовых разрядов сумматора. Для осуществления сдвига данную микрооперацию необходимо выполнять одновременно с 18-й
24	Разрешить перенос/сдвиг влево	Данная микрооперация имеет два различных назначения. Первое – разрешение переноса влево (в старший числовой разряд) при выполнении сумматором операции сложения. Второе – сдвиг влево на один разряд содержимого числовых разрядов сумматора. Для осуществления сдвига данную микрооперацию необходимо выполнять одновременно с 18й
25	Сброс ТЗН RGB	Сброс (установка в ноль) содержимого триггера знака (знакового разряда) регистра RGB
26	Прием ТЗН RGB	Прием из памяти либо сумматора содержимого знакового разряда в знаковый разряд регистра RGB
27	Инверсия ТЗН RGB	Инверсия (смена значения на противоположное) содержимого знакового разряда регистра RGB
28	Выдача ТЗН RGB	Выгрузка содержимого знакового разряда регистра RGB (для передачи в сумматор)
29	Выдача RGB в обратном коде	Выгрузка содержимого регистра RGB, преобразованного в обратный код (для передачи, например, в сумматор). Данная микрооперация необходима для выполнения операций над отрицательными числами, поскольку сумматор выполняет вычисления в обратных кодах, а для отрицательных чисел обратный код отличается от прямого. При выполнении операций над положительными числами эту микрооперацию можно не использовать (обратный код совпадает с прямым)
30	Сброс числовых разрядов RGB	Сброс (установка в ноль) содержимого числовых разрядов регистра RGB
31	Прием числовых разрядов RGB	Прием значений в числовые разряды регистра RGB
32	Установ младшего разряда RGB в 1	Установка содержимого младшего числового разряда регистра RGB в единицу
33	Сдвиг влево RGB	Сдвиг содержимого числовых разрядов регистра RGB влево на один разряд
34	Выдача RGB	Выгрузка содержимого числовых разрядов регистра RGB (для передачи, например, в сумматор)
35	Разрешить чтение из "памяти"	Позволяет считывать содержимое памяти для приема в регистры (выгрузка из памяти в сумматор напрямую, минуя регистры, не поддерживается)

4.2 Модель устройства управления

Модель устройства управления представляет собой таблицу (рис. 3), строки которой соответствуют адресам в памяти, а столбцы – содержимому микрокоманд (нумерация столбцов идет справа налево). Микрокоманда состоит из двух частей – операционной (задает набор микроопераций, выполняемых данной микрокомандой) и адресной (работа с условиями и адрес перехода на следующую микрокоманду). Запись программы в память осуществляется путем записи микрокоманд в ячейки памяти посредством активации (с помощью левой кнопки мыши) ячеек таблицы, соответствующих нужным действиям (для операционной части), а также условиям и адресам (для адресной части микрокоманды):

- микрооперации (поля с 1 по 35 задают соответствующие им по номерам микрокоманды);
- поле с номером 36 соответствует кнопке «Выполнить» модели АЛУ;
- поле 37 – уменьшение содержимого счетчика на единицу;
- номер проверяемого условия (поля 38-40);
- адрес следующей микрокоманды (поля 41-45 и 46-50) в случае использования принудительной системы адресации.

Запись программы в память может начинаться с любого адреса, не обязательно с первого, но начальный адрес (адрес первой микрокоманды) должен быть загружен в регистр адреса микрокоманды (РАМК), поскольку он показывает с какого адреса устройство управления начнет выполнять загруженную программу.

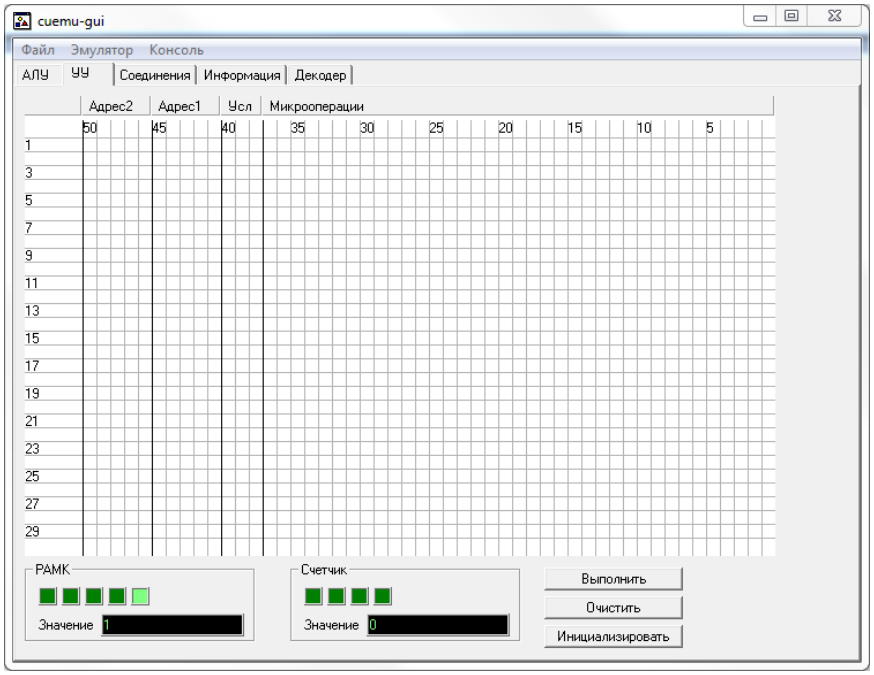

Рис. 3. Модель устройства управления

Устройство управления позволяет реализовывать циклы (для этого используется счетчик, значение в который загружается вручную в двоичном коде), условные переходы (поля 38-40 задают номер условия, записанный в двоичном коде, количество поддерживаемых условий равняется 7) и ветвления. Кроме того, устройство управления может работать в разных системах адресации, как естественной, так и принудительной (поля 41-45 и 46-50 позволяют задавать адреса перехода при использовании принудительной системы адресации), переключение (рис. 4) осуществляется в пункте меню «Эмулятор» (по умолчанию при запуске программы и при загрузке сохраненного файла всегда включена естественная адресация). Более подробно о системах адресации написано в пункте 4.3 данных методических указаний.

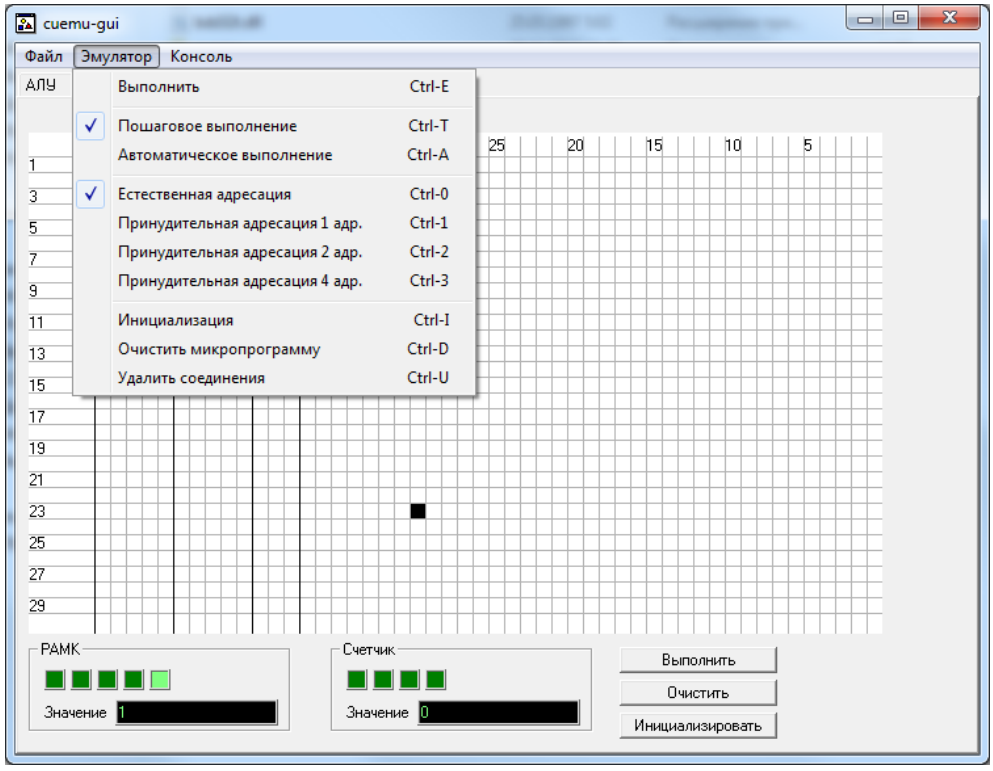

Рис. 4. Поддерживаемые системы адресации и их переключение

4.3 Системы адресации микрокоманд

Естественная адресация. При использовании данной системы адресации адрес следующей микрокоманды получается путем прибавления единицы к адресу предыдущей микрокоманды. Таким образом, программа выполняется последовательно от первого адреса до последнего. Циклы, условные переходы и ветвления при использовании данной системы адресации реализовать невозможно.

Принудительная адресация с одним адресным полем. При использовании данной системы адресации в поле «Адрес 1» (41-45 поля)

может быть задан адрес условного перехода. В том случае, если для микрокоманды логическое условие в полях 38-40 не задано либо при его проверке оказалось равным нулю, то после выполнения текущей микрокоманды произойдет переход на следующий адрес (+1 относительно текущего адреса). При равенстве логического условия единицы происходит переход по адресу, указанному в поле «Адрес 1».

Принудительная адресация с двумя адресными полями. При использовании данной системы адресации в полях «Адрес 1» (41-45 поля) и «Адрес 2» могут быть заданы адреса условного перехода. В том случае, если для микрокоманды логическое условие в полях 38-40 не задано либо при его проверке оказалось равным нулю, то после выполнения текущей микрокоманды произойдет переход на адрес, указанный в поле «Адрес 1». При равенстве логического условия единицы происходит переход по адресу, указанному в поле «Адрес 2».

Список литературы

1. Таненбаум Э. Архитектура компьютера. 5-е издание. Спб.: Питер, 2007.

1. Цилькер Б.Я. Организация ЭВМ и систем. Спб.: Питер, 2007.

2. Жмакин А.П. Архитектура ЭВМ. Спб.: БХВ-Петербург, 2008.

3. Новожилов О.П. Архитектура ЭВМ и систем. М.: Юрайт, 2012.

4. Баула В.Г. Архитектура ЭВМ и операционные среды. М.: Академия, 2012.