НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**КУРСОВА РОБОТА**

**з дисципліни "Комп’ютерна логіка"**

**на тему: Пристрій виконання арифметичної операції**

Виконав: Давидчук А. М.

Група: ІО-41

Варіант: 4106

Допущений до захисту \_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оцінка, підпис керівника)

Київ – 2025 р.

**Опис альбому**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *№ рядка* | *Формат* | | *Позначення* | | | | | *Найменування* | | | | | *Кіл-сть* | *Примітка* |
| *1* |  | |  | | | | | Документація загальна | | | | |  |  |
| *2* |  | |  | | | | |  | | | | |  |  |
| *3* |  | |  | | | | | Наново розроблена | | | | |  |  |
| *4* |  | |  | | | | |  | | | | |  |  |
| *5* | *А4* | | КПІ.ФІОТ.ІО4106.001 ОА | | | | | Курсова робота  Опис альбому | | | | | 1 |  |
| *6* |  | |  | | | | |  | | | | |  |  |
| *7* | *А4* | | КПІ.ФІОТ.ІО4106.002 Т3 | | | | | Курсова робота  Технічне завдання | | | | | 5 |  |
| *8* |  | |  | | | | |  | | | | |  |  |
| *9* |  | |  | | | | |  | | | | |  |  |
| *10* | *А2* | | КПІ.ФІОТ.ІО4106.003 Е2 | | | | | Автомат керуючий.  Схема електрична  функціональна | | | | | 1 |  |
| *11* |  | |  | | | | |  | | | | |  |  |
| *12* | *А4* | | КПІ.ФІОТ.ІО4106.004 ПЗ | | | | | Курсова робота  Пояснювальна записка | | | | | 46 |  |
| *13* |  | |  | | | | |  | | | | |  |  |
| *14* |  | |  | | | | |  | | | | |  |  |
| *15* |  | |  | | | | |  | | | | |  |  |
| *16* |  | |  | | | | |  | | | | |  |  |
| *17* |  | |  | | | | |  | | | | |  |  |
| *18* |  | |  | | | | |  | | | | |  |  |
| *19* |  | |  | | | | |  | | | | |  |  |
| *20* |  | |  | | | | |  | | | | |  |  |
| *21* |  | |  | | | | |  | | | | |  |  |
| *22* |  | |  | | | | |  | | | | |  |  |
| *23* |  | |  | | | | |  | | | | |  |  |
| *24* |  | |  | | | | |  | | | | |  |  |
| *25* |  | |  | | | | |  | | | | |  |  |
|  | |  | |  |  |  | ***КПІ.ФІОТ.ІО4106.001 ОА*** | | | | | | | |
|  | |  | |  |  |  |
| *Зм.* | | *Арк.* | | *№ докум* | *Підпис* | *Дата* |
| *Розробив* | | | | *Давидчук А. М.* |  |  | *КУРСОВА РОБОТА*  *Опис альбому* | | *Літера* | | | *Аркуш* | | *Аркушів* |
| *Перевірив* | | | | *Верба О. А.* |  |  |  |  |  | *1* | | *1* |
| *Реценз.* | | | |  |  |  | ***НТУУ «КПІ» ФІОТ***  ***Група ІО-41*** | | | | | |
| *Н. контр.* | | | |  |  |  |
| *Затверд.* | | | | *Жабін В. І.* |  |  |

**Технічне завдання**

**Зміст**

[***1.*** ***ПРИЗНАЧЕННЯ РОЗРОБЛЮВАНОГО ОБ’ЄКТУ*** 2](#_Toc167457519)

[***2.*** ***ВИХІДНІ ДАНІ ДЛЯ РОЗРОБКИ*** 2](#_Toc167457520)

[***3.*** ***СКЛАД ПРИСТРОЇВ*** 4](#_Toc167457521)

[***4.*** ***ЕТАПИ І ТЕРМІНИ ПРОЕКТУВАННЯ***  4](#_Toc167457522)

[***5.*** ***ПЕРЕЛІК ТЕКСТОВОЇ І ГРАФІЧНОЇ ДОКУМЕНТАЦІЇ***  5](#_Toc167457523)

1. ***ПРИЗНАЧЕННЯ РОЗРОБЛЮВАНОГО ОБ’ЄКТУ***

У курсовій роботі необхідно виконати арифметичні операції з числами у форматі з плаваючою комою.

Метою таких операцій є дослідження та визначення методів, за допомогою яких ці операції реалізуються у комп’ютері. Ці розділи допоможуть зрозуміти, як комп’ютер виконує обчислення, такі як множення, ділення, додавання, віднімання тощо. Вивчення цих процесів є важливим для розробки та оптимізації арифметичних операцій у комп’ютерах, що, в свою чергу, сприяє підвищенню їх продуктивності та ефективності.

1. ***ВИХІДНІ ДАНІ ДЛЯ РОЗРОБКИ***

Варіант індивідуального завдання визначається вихідними значеннями операндів X і Y, які визначаються номером групи і номером студента у групі наступним чином. Перевести номер варіанту в двійкову систему.

Номер залікової книжки

Записати два 10-розрядних двійкових числа:

X = і Y = .

Числа X і Y в прямому коді записати у формі з плаваючою комою у класичному варіанті (з незміщеним порядком і повною мантисою). На мантису відвести 6 основних розрядів, на порядок студент самостійно вибирає необхідну кількість розрядів, щоб не було переповнення порядку при виконання чергової операції, один знаковий розряд для мантиси та один знаковий розряд для порядку числа.

Виконати 8 операцій з числами, що подані з плаваючою комою в класичному варіанті: чотири способи множення, два способи ділення, додавання та обчислення кореня додатного числа. Операндами для першого способу множення є задані числа X та Y. Для кожної наступної операції першим операндом є результат попередньої операції, а другим операндом завжди є число Y (для операції обчислення кореня операндом є результат додавання зі знаком плюс).

Перед виконанням чергової операції треба представити необхідні операнди в класичному вигляді з порядком та мантисою.

Для операції з номером  додатково виконати:

1. побудувати на базі операторної схеми функціональну схему з відображенням управляючих сигналів, входів для запису операндів при ініціалізації пристрою і схем формування внутрішніх логічних умов; схему подати як рисунок у пояснювальній записці.
2. розробити структурний мікроалгоритм (мікрооперації замінюються управляючими сигналами виду W, SL, SR тощо), а також закодований мікроалгоритм (сигнали, що завжди формуються разом, позначаються одним символом);
3. якщо в отриманому мікроалгоритмі в двох суміжних операторних вершинах виконуються різні мікрооперації на одному і тому же регістрі/лічильнику, то між вказаними операторними вершинами треба додати порожню вершину; при цьому різні управляючи сигнали на регістрі не будуть перекриватися в часі, тобто не виникне збій в роботі пристрою;
4. для операції з парним номером подати граф управляючого автомата Мура з відображенням кодів вершин, а для непарного номера – граф автомата Мілі;
5. побудувати управляючий автомат на тригерах та елементах булевого базису. Відповідно із кодом x2x1, що приймає значення 00, 01, 10, 11 вибрати відповідно JK-, RS-, T-, D-тригери для побудови автомата; виконати сумісну мінімізацію функцій управління тригерами та вихідних сигналів;
6. у форматі А2 або А3 виконати креслення функціональної схеми управляючого автомата з урахуванням діючих стандартів для схем Е2.
7. у моделюючій програмі ПРОГМОЛС (AFDK) побудувати операційний пристрій і керуючий автомат виконання даної операції. Надати часову діаграму роботи автомата. Виконати числовий приклад за варіантом. Зробити скріншот побудованої схеми з числовим результатом.
8. ***СКЛАД ПРИСТРОЇВ***

**Керуючий цифровий автомат:**

Керуючий автомат містить у собі комбінаційну схему (КС) і пам’ять, що складається з тригерів. Входами КС є виходи тригерів і логічні умови, входами тригерів є логічні умови.

**Операційний пристрій:**

Операційний пристрій містить у собі регістри та суматори для виконання заданих операцій і керується цифровим автоматом. Входами і виходами операційного пристрою є надані операнди. Також операційний пристрій задає логічні умови для керуючого автомата.

1. ***ЕТАПИ ПРОЕКТУВАННЯ І ТЕРМІНИ ВИКОНАННЯ***
2. Дослідження способів виконання операцій:
   1. теоретичне обґрунтування способу виконання операції;
   2. операційну схему;
   3. змістовний мікроалгоритм виконання операції;
   4. таблицю станів регістрів (лічильника), довжина яких забезпечує одержання 6 основних розрядів мантиси результату;
   5. обробку порядків в довільній формі;
   6. форму запису нормалізованого результату з плаваючою комою в пам’ять комп’ютера в прямому коді;
   7. форму запису нормалізованого результату з плаваючою комою за стандартом ANSI/IEEE 754-2008 в короткому форматі.
3. Для операції з номером x3x2x1 додатково зробити:
   1. функціональну схему;
   2. структурний мікроалгоритм (з урахуванням порожніх вершин між вершинами, у яких дії виконуються на одному регістрі);
   3. для операції з парним номером подати граф управляючого автомата Мура з відображенням кодів вершин, а для непарного номера – граф автомата Мілі;
   4. побудувати управляючий автомат на тригерах та елементах булевого базису. Відповідно із кодом , що приймає значення 00, 01, 10, 11, вибрати відповідно JK-, RS-, T-, D-тригери для побудови автомата;
   5. у форматі А2 або А3 виконати креслення функціональної схеми управляючого автомата з урахуванням діючих стандартів для схем Е2.
   6. у моделюючій програмі ПРОГМОЛС (AFDK) побудувати операційний пристрій і керуючий автомат виконання даної операції. Надати часову діаграму роботи автомата. Виконати числовий приклад за варіантом. Зробити скріншот побудованої схеми з числовим результатом.
4. ***ПЕРЕЛІК ТЕКСТОВОЇ І ГРАФІЧНОЇ ДОКУМЕНТАЦІЇ***

* Титульний лист;
* Сторінка з написом “Опис альбому”;
* Опис альбому;
* Сторінка з написом “Технічне завдання”;
* Технічне завдання;
* Сторінка з написом “Автомат керуючий. Схема електрична функціональна”;
* Керуючий автомат, схема електрична функціональна;
* Сторінка з написом “Пояснювальна записка”;
* Пояснювальна записка.

**Автомат керуючий.**

**Схема електрична функціональна**

**Пояснювальна записка**

**Зміст**

[***1******ВСТУП*** *2*](#_Toc167453472)

[***2******ОПЕРАНДИ НА КУРСОВУ РОБОТУ*** *2*](#_Toc167453473)

[***3******ДОСЛІДЖЕННЯ СПОСОБІВ ВИКОНАННЯ ОПЕРАЦІЙ*** *3*](#_Toc167453474)

[*3.1 Перший спосіб множення 3*](#_Toc167453475)

[*3.2 Другий спосіб множення 7*](#_Toc167453476)

[*3.3 Третій спосіб множення 11*](#_Toc167453477)

[*3.4 Четвертий спосіб множення 15*](#_Toc167453478)

[*3.5 Перший спосіб ділення 19*](#_Toc167453479)

[*3.6 Другий спосіб ділення 23*](#_Toc167453480)

[*3.7 Додавання чисел ДК 27*](#_Toc167453481)

[*3.8 Добування кореню додатного числа 29*](#_Toc167453482)

[***4******ПОБУДОВА ФУНКЦІОНАЛЬНИХ СХЕМ*** *33*](#_Toc167453483)

[*4.1 Функціональна схема 33*](#_Toc167453484)

[*4.2 Змістовний мікроалгоритм 34*](#_Toc167453485)

[*4.3 Структурний мікроалгоритм 35*](#_Toc167453486)

[*4.4 Автомат Мура 36*](#_Toc167453487)

[***5******ВИСНОВОК*** *42*](#_Toc167453488)

[***6******СПИСОК ЛІТЕРАТУРИ*** *44*](#_Toc167453489)

1. ***ВСТУП***

Курсова робота виконана за номером технічного завдання , і складається з двох частин: дослідження операційних способів виконання операцій та побудова функціональних схем. Вихідними даними при дослідженні є результат операцій. Вхідними даними при синтезі автомата є заданий алгоритм, тип тригера та елементна база.

1. ***ОПЕРАНДИ НА КУРСОВУ РОБОТУ***

Операнди: X = -001010,0110; Y = +010000,1010

Для операції з номером виконати четвертий пункт. Також подати граф Мура, оскільки варіант парний. Відповідно до автомат побудувати на Т-тригерах. На порядок буду виділяти 5 біт щоб уникнути переповнення при виконанні операцій.

Коди чисел в ПК:

X = 1.001010,0110

Y = 0.010000,1010

Порядок чисел: ;

Не нормалізовані мантиси чисел (10 розрядів): ;

Нормалізовані мантиси чисел (10 розрядів): ;

Оновлені порядки нормалізованих мантис:

;

Округлені до 6 розрядів мантиси чисел: *;*

Для округлення відбулось шляхом додавання 1 до 7 біту з циклічним переносом (через те, що 7 біт = 1), а для зберігаючи тільки перші 6 бітів (7 біт = 0).

Числа у класичному форматі плаваючої коми:

X = 0 | 00100 | 1 | 101010

Y = 0 | 00101 | 0 | 100001

Числа X та Y за стандартом ANSI/IEEE 754-2008 в короткому 32-розрядному форматі:

1. ***ДОСЛІДЖЕННЯ СПОСОБІВ ВИКОНАННЯ ОПЕРАЦІЙ***
   1. *Перший спосіб множення*

Операнди: X = 0 | 00100 | 1 | 101010; Y = 0 | 00101 | 0 | 100001;

* + 1. *Теоретичне обґрунтування способу виконання операцій*

Робота з мантисами чисел: при множенні за першим методом у першому такті кожного *i*-го циклу відбувається перевірка значення наймолодшого (-го) біту регістру RG2, який містить наступну цифру множника. Якщо цей біт дорівнює 1, то вміст регістру RG3 додається до активної суми часткових добутків, що зберігаються в регістрі RG1. В іншому випадку, якщо цей біт дорівнює 0, додавання не відбувається. У другому такті реалізується правий зсув у регістрах RG1 і RG2, який можна прирівняти до множення їхнього вмісту на . Під час цього зсуву, наймолодший біт з регістру RG1 переноситься в вільний старший біт регістру RG2. Після проведення таких циклів, молодші біти кінцевого добутку з довжиною бітів зберігаються в регістрі RG2, тоді як старші біти знаходяться в RG1.

Робота з порядками чисел: порядок результуючого числа є сумою порядків операндів.

* + 1. *Операційна схема*

Изображение выглядит как диаграмма, текст, План, Технический чертеж

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 – Операційна схема першого способу множення

* + 1. *Змістовний мікроалгоритм*

Изображение выглядит как текст, снимок экрана, Шрифт, белый

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 – Змістовний мікроалгоритм першого способу множення

* + 1. *Таблиця станів регістрів множення 1 способом*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № |  |  |  | CT | Мікрооперації |
| – | 0000000 | 10101**0** | 0100001 | 110 | ;; |
| 1 | 0000000  >>0000000 | 101010  >>01010**1** | 101 | RG1 = 0.r(RG1);  RG2 = RG1[1].r(RG2);  CT = CT – 1; |
| 2 | 0000000 + 0100001  0100001  >>0010000 | 001010  >>10101**0** | 100 | RG1 = RG1 + RG3;  RG1 = 0.r(RG1);  RG2 = RG1[1].r(RG2);  CT = CT – 1; |
| 3 | 0010000  >>0001000 | 101010  >>01010**1** | 011 | RG1 = 0.r(RG1);  RG2 = RG1[1].r(RG2);  CT = CT – 1; |
| 4 | 0001000 + 0100001  0101001  >>0010100 | 010101  >>10101**0** | 010 | RG1 = RG1 + RG3;  RG1 = 0.r(RG1);  RG2 = RG1[1].r(RG2);  CT = CT – 1; |
| 5 | 0010100  >>0001010 | 101010  >>01010**1** | 001 | RG1 = 0.r(RG1);  RG2 = RG1[1].r(RG2);  CT = CT – 1; |
| 6 | 0001010 + 0100001  0101011  >>0010101 | 010101  >>101010 | **000** | RG1 = RG1 + RG3;  RG1 = 0.r(RG1);  RG2 = RG1[1].r(RG2);  CT = CT – 1; |

Результат множення мантис (7 розрядів): ,0101011

* + 1. *Обробка порядків*

* + 1. *Форма запису нормалізованого результату*

Результат множення мантис: ,0101011

Враховуючи знаки мантис 0 ⊕ 1 = 1 отримаємо:

1,0101011

Отже, нормалізована мантиса має вигляд: 1,1010110; Порядок 01000

Помітимо, що 7 біт = 0, тому округлення можна провести шляхом збереження перших 6 бітів.

Нормалізований результат: Z = 0 | 01000 | 1 | 101011

* + 1. *Короткий формат за стандартом ANSI/IEEE 754-2008*

Число Z за стандартом ANSI/IEEE 754-2008 в короткому 32-розрядному форматі:

* 1. *Другий спосіб множення*

Операнди: Z = 0 | 01000 | 1 | 101011; Y = 0 | 00101 | 0 | 100001

* + 1. *Теоретичне обґрунтування способу виконання операцій*

Робота з мантисами чисел: перед тим як розпочати множення другим методом, множник X заносять до регістру RG2, тоді як множене Y розміщують у молодших бітах регістру RG3, встановлюючи Y0 як . В кожному i-му циклі множення рішення про додавання вмісту регістрів RG3 і RG1 базується на значенні біта RG2[1]. При цьому у регістрі RG3 проводять зсув вліво на один біт, що дозволяє отримати нову величину , яка дорівнює подвоєному попередньому значенню . Результат множення знаходиться в регістрі RG1.

Робота з порядками чисел: порядок результуючого числа є сумою порядків операндів.

* + 1. *Операційна схема*

Изображение выглядит как текст, диаграмма, линия, Технический чертеж

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 – Операційна схема другого способу множення

* + 1. *Змістовний мікроалгоритм*

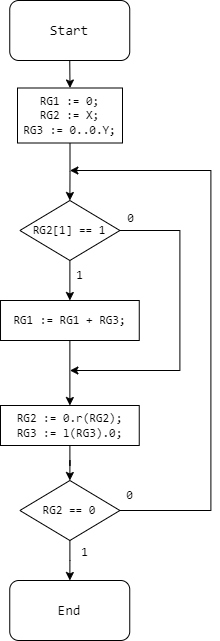


Рисунок 4 – Змістовний мікроалгоритм другого способу множення

* + 1. *Таблиця станів регістрів множення 2 способом*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | RG1 | RG2 | RG3 | Мікрооперації |
| – | 000000000000 | 10101**1** | 000000100001 | RG1 = 0; RG2 = X;  RG3 = 0..0.Y |
| 1 | 000000000000 + 000000100001 000000100001 | 101011 01010**1** | 000000100001 000001000010 | RG1 = RG1 + RG3;  RG2 = 0.r(RG2);  RG3 = l(RG3).0; |
| 2 | 000000100001 + 000001000010 000001100011 | 010101 00101**0** | 000001000010 000010000100 | RG1 = RG1 + RG3;  RG2 = 0.r(RG2);  RG3 = l(RG3).0; |
| 3 | 000001100011 | 001010 00010**1** | 000010000100 000100001000 | RG2 = 0.r(RG2);  RG3 = l(RG3).0; |
| 4 | 000001100011 + 000100001000 000101101011 | 000101 00001**0** | 000100001000 001000010000 | RG1 = RG1 + RG3;  RG2 = 0.r(RG2);  RG3 = l(RG3).0; |
| 5 | 000101101011 | 000010 00000**1** | 001000010000 010000100000 | RG2 = 0.r(RG2);  RG3 = l(RG3).0; |
| 6 | 000101101011 + 010000100000 010110001011 | 000001 **000000** | 010000100000 100001000000 | RG1 = RG1 + RG3;  RG2 = 0.r(RG2);  RG3 = l(RG3).0; |

Результат множення мантис (7 розрядів): ,0101100

* + 1. *Обробка порядків*

* + 1. *Форма запису нормалізованого результату*

Результат множення мантис: ,0101100

Враховуючи знаки мантис отримаємо: 0 ⊕ 1 = 1

*=* 1,0101100

Отже, нормалізована мантиса має вигляд: 1,1011000; Порядок = 01100.

Помітимо, що 7 біт = 0, тому округлення можна провести шляхом збереження перших 6 бітів.

Нормалізований результат: N = 0 | 01100 | 1 | 101100

* + 1. *Короткий формат за стандартом ANSI/IEEE 754-2008*

Число Z за стандартом ANSI/IEEE 754-2008 в короткому 32-розрядному форматі:

= 1.10001010,01100000000000000000000

* 1. *Третій спосіб множення*

Операнди: N = 0 | 01100 | 1 | 101100; Y = 0 | 00101 | 0 | 100001

* + 1. *Теоретичне обґрунтування способу виконання операцій*

Робота з мантисами чисел: у третьому методі множення множник X записують у старші біти регістру RG2, причому біт RG2[1] встановлюють у стан "0". Вага молодшого біту RG3 визначається як , тому значення в регістрі RG3 представляє собою . В кожному циклі множення умовою для виконання підсування є значення RG2[n+1], яке має бути дорівнює 1. В регістрах RG1 і RG2 проводять лівий зсув. В результаті цього, додаванням вмісту регістрів RG3 і RG1 може виникнути перенос у молодший розряд регістру RG2, що обробляється за допомогою суматора з переносом (SM). Збільшення довжини регістру RG2 на один біт запобігає поширенню переносу на розряди множника. Після завершення n циклів множення молодші біти кінцевого добутку будуть розташовані в регістрі RG1, а старші біти — в регістрі RG2.

Робота з порядками чисел: порядок результуючого числа є сумою порядків операндів.

* + 1. *Операційна схема*

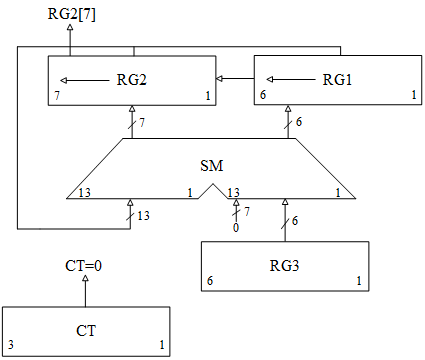


Рисунок 5 – Операційна схема третього способу множення

* + 1. *Змістовний мікроалгоритм THINK*

Изображение выглядит как текст, снимок экрана, Шрифт, черно-белый

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6 – Змістовний мікроалгоритм третього способу множення

* + 1. *Таблиця станів регістрів 3 способу множення*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | RG2 | RG1 | RG3 |  | Мікрооперації |
| – | **1**011000 | 000000 | 100001 | 110 | RG1 = 0; RG2 = X.0; RG3 = Y; CT = 6; |
| 1 | 1011000 **0**110001 | 000000 + 100001 100001 000010 | 101 | RG1 = RG1 + RG3;  RG2 = RG2 + 0 + CL;  RG2 = l(RG2).RG1[6];  RG1 = l(RG1).0; CT = CT – 1; |
| 2 | 0110001 **1**100010 | 000010 000100 | 100 | RG2 = l(RG2).RG1[6];  RG1 = l(RG1).0; CT = CT – 1; |
| 3 | 1100010 **1**000101 | 000100 + 100001 100101 001010 | 011 | RG1 = RG1 + RG3;  RG2 = RG2 + 0 + CL;  RG2 = l(RG2).RG1[6];  RG1 = l(RG1).0; CT = CT – 1; |
| 4 | 1000101 **0**001011 | 001010 + 100001 101011 010110 | 010 | RG1 = RG1 + RG3;  RG2 = RG2 + 0 + CL;  RG2 = l(RG2).RG1[6];  RG1 = l(RG1).0; CT = CT – 1; |
| 5 | 0001011 **0**010110 | 010110 101100 | 001 | RG2 = l(RG2).RG1[6];  RG1 = l(RG1).0; CT = CT – 1; |
| 6 | 0010110 0101101 | 101100 011000 | **000** | RG2 = l(RG2).RG1[6];  RG1 = l(RG1).0; CT = CT – 1; |

Результат множення мантис (7 розрядів): ,0101101

* + 1. *Обробка порядків*

* + 1. *Форма запису нормалізованого результату*

Результат множення мантис: ,0101101

Враховуючи знаки мантис отримаємо: 0 ⊕ 1 = 1

= 1,0101101

Отже, нормалізована мантиса має вигляд: 1,1011010; Порядок = 10000.

Помітимо, що 7 біт = 0, тому округлення можна провести шляхом збереження перших 6 бітів.

Нормалізований результат: P = 0 | 10000 | 1 | 101101

* + 1. *Короткий формат за стандартом ANSI/IEEE 754-2008*

Число Z за стандартом ANSI/IEEE 754-2008 в короткому 32-розрядному форматі:

P = 1.10001110,01101000000000000000000

* 1. *Четвертий спосіб множення*

Операнди: P = 0 | 10000 | 1 | 101101; Y = 0 | 00101 | 0 | 100001

* + 1. *Теоретичне обґрунтування способу виконання операцій*

Робота з мантисами чисел: перед початком множення четвертим методом, множник записують у регістр RG2, тоді як множене Y — у старші біти регістру RG3, встановлюючи . У кожному циклі множення старший біт регістру RG2 контролює процес додавання, в той час як у регістрі RG3 проводять правий зсув на один розряд. Цей зсув ефективно відповідає множенню вмісту регістру на . Результат множення формується в регістрі RG1.

Робота з порядками чисел: порядок результуючого числа є сумою порядків операндів.

* + 1. *Операційна схема*

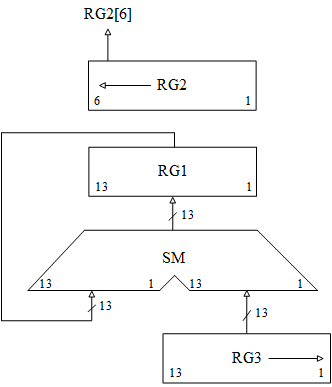


Рисунок 7 – Операційна схема четвертого способу множення

* + 1. *Змістовний мікроалгоритм*

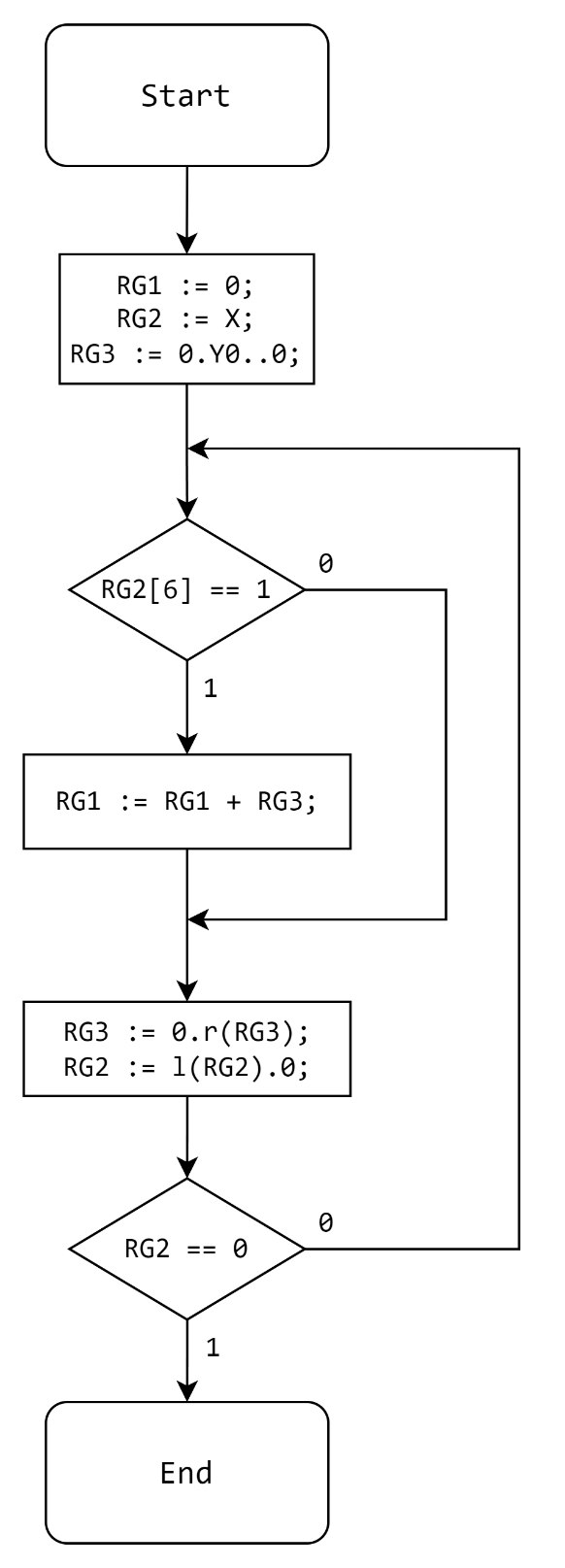


Рисунок 8 – Змістовний мікроалгоритм четвертого способу множення

* + 1. *Таблиця станів регістрів 4 способу множення*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | RG1 | RG2 | RG3 | Мікрооперації |
| - | 0000000000000 | **1**01101 | 0100001000000 | RG1 = 0; RG2 = X;  RG3 = 0.Y0..0; |
| 1 | 0000000000000 + 0100001000000 0100001000000 | 101101 **0**11010 | 0100001000000 0010000100000 |  |
| 2 | 0100001000000 | 011010 **1**10100 | 0010000100000 0001000010000 |  |
| 3 | 0100001000000 + 0001000010000 0101001010000 | 110100 **1**01000 | 0001000010000 0000100001000 |  |
| 4 | 0101001010000 + 0000100001000 0101101011000 | 101000 **0**10000 | 0000100001000 0000010000100 |  |
| 5 | 0101101011000 | 010000 **1**00000 | 0000010000100 0000001000010 |  |
| 6 | 0101101011000 + 0000001000010 0101110011010 | 100000 000000 | 0000001000010 0000000100001 |  |

Результат множення мантис (7 розрядів): ,0101110

* + 1. *Обробка порядків*

* + 1. *Форма запису нормалізованого результату*

Результат множення мантис: ,0101110

Враховуючи знаки мантис отримаємо: 0 ⊕ 1 = 1

= 1,0101110

Отже, нормалізована мантиса має вигляд: 1,1011100; Порядок = 10100.

Помітимо, що 7 біт = 0, тому округлення можна провести шляхом збереження перших 6 бітів.

Нормалізований результат: A = 0 | 10100 | 1 | 101110

* + 1. *Короткий формат за стандартом ANSI/IEEE 754-2008*

Число Z за стандартом ANSI/IEEE 754-2008 в короткому 32-розрядному форматі:

A = 1.10010010,01110000000000000000000

* 1. *Перший спосіб ділення*

Операнди: A = 0 | 10100 | 1 | 101110; Y = 0 | 00101 | 0 | 100001

* + 1. *Теоретичне обґрунтування способу виконання операцій*

Робота з мантисами чисел: під час виконання ділення за першим методом відбувається зсув вліво залишку, при цьому дільник залишається незмінним. Черговий залишок формується в регістрі RG2 (у вихідному стані в регістр RG2 записується значення X), із якого безпосередньо виводиться інформація на входи суматора SM, тобто додаткові ланцюги для передачі коду не потрібні. Дільник Y розташований у регістрі RG1. Процес ділення відбувається протягом (n+1) циклів, а результат зберігається у регістрі RG3. Розряд RG3[n+1] використовується для ідентифікації завершення операції, що позначається маркерним нулем на виході цього біта.

Робота з порядками чисел: порядок результуючого числа є різницею порядків операндів.

* + 1. *Операційна схема*

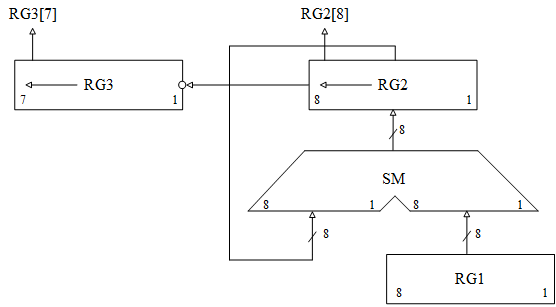


Рисунок 9 – Операційна схема першого способу ділення

* + 1. *Змістовний мікроалгоритм*

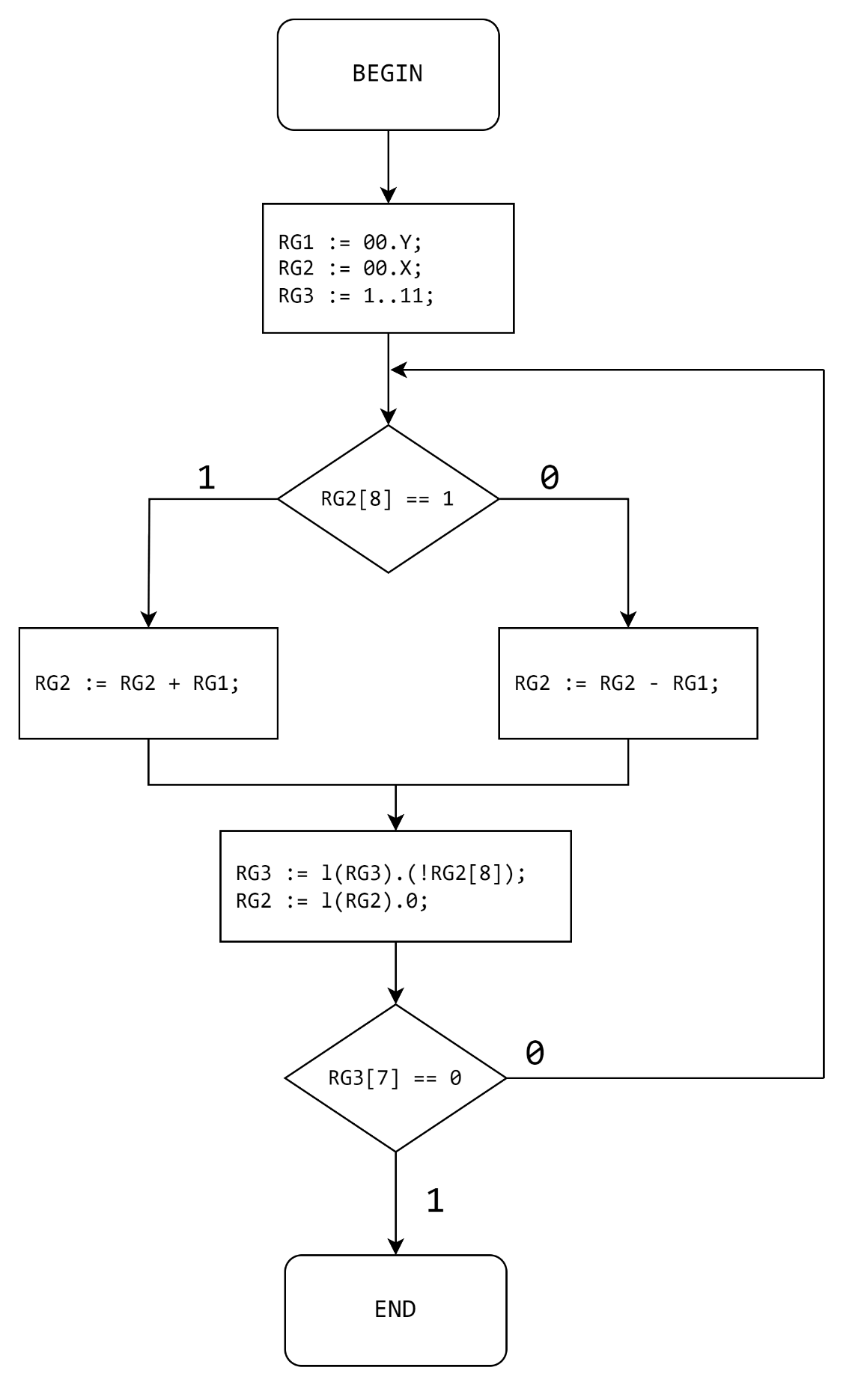


Рисунок 10 – Змістовний мікроалгоритм першого способу ділення

* + 1. *Таблиця станів регістрів 1 способу ділення*

Оскільки , то потрібно зменшити , шляхом її зсуву на один розряд вправо, тим самим додавши до одиницю. Звідси = 10101, = 010111, тоді і тепер можемо виконувати ділення

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | RG3 | RG2 | RG1 | Мікрооперації |
| - | 1111111 | **0**0010111 | RG1 = 00100001  -RG1 + 1 (-Y в ДК): 11011111 | RG1 = 00.Y; RG2 = 00.X;  RG3 = 1..11; |
| 1 | 1111111 1111110 | 00010111 + 11011111 11110110 **1**1101100 | RG2 = RG2 – RG1;  RG3 = l(RG3).(!RG2[8]);  RG2 = l(RG2).0; |
| 2 | 1111110 1111101 | 11101100 + 00100001 00001101 **0**0011010 | RG2 = RG2 + RG1;  RG3 = l(RG3).(!RG2[8]);  RG2 = l(RG2).0; |
| 3 | 1111101 1111010 | 00011010 + 11011111 11111001 **1**1110010 | RG2 = RG2 – RG1;  RG3 = l(RG3).(!RG2[8]);  RG2 = l(RG2).0; |
| 4 | 1111010 1110101 | 11110010 + 00100001 00010011 **0**0100110 | RG2 = RG2 + RG1;  RG3 = l(RG3).(!RG2[8]);  RG2 = l(RG2).0; |
| 5 | 1110101 1101011 | 00100110 + 11011111 00000101 **0**0001010 | RG2 = RG2 – RG1;  RG3 = l(RG3).(!RG2[8]);  RG2 = l(RG2).0; |
| 6 | 1101011 1010110 | 00001010 + 11011111 11101001 **1**1010010 | RG2 = RG2 – RG1;  RG3 = l(RG3).(!RG2[8]);  RG2 = l(RG2).0; |
| 7 | 1010110 0101100 | 11010010 + 00100001 11110011 11100110 | RG2 = RG2 + RG1;  RG3 = l(RG3).(!RG2[8]);  RG2 = l(RG2).0; |

Результат ділення мантис (6 розрядів): ,101100

* + 1. *Обробка порядків*

* + 1. *Форма запису нормалізованого результату*

Результат ділення мантис: ,101100

Враховуючи знаки мантис отримаємо: 0 ⊕ 1 = 1

= 1,101100 (мантиса вже є нормалізованою)

Отже, нормалізована мантиса має вигляд: 1,101100

Нормалізований результат: B = 0 | 10000 | 1 | 101100

* + 1. *Короткий формат за стандартом ANSI/IEEE 754-2008*

Число Z за стандартом ANSI/IEEE 754-2008 в короткому 32-розрядному форматі:

B = 1.10001110,01100000000000000000000

* 1. *Другий спосіб ділення*

Операнди: B = 0 | 10000 | 1 | 101100; Y = 0 | 00101 | 0 | 100001

* + 1. *Теоретичне обґрунтування способу виконання операцій*

Робота з мантисами чисел: під час реалізації ділення другим методом, який передбачає зсув дільника, відбувається збільшення розрядності регістрів RG1, RG3 та суматора SM. Залежно від значення знакового біта RG2[2n+1], до вмісту регістра RG2 додається або віднімається вміст регістра RG1. Віднімання реалізується шляхом представлення від’ємного числа регістра RG1 в ДК (інверсії коду з RG1 та додавання одиниці на вхід переносу суматора SM (D=1)). Наступна цифра результату ділення заноситься у регістр RG3 із виходу переносу суматора під час зсуву.

Робота з порядками чисел: порядок результуючого числа є різницею порядків операндів.

* + 1. *Операційна схема*

Изображение выглядит как диаграмма, текст, План, Технический чертеж

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 11 – Операційна схема другого способу ділення

* + 1. *Змістовний мікроалгоритм*

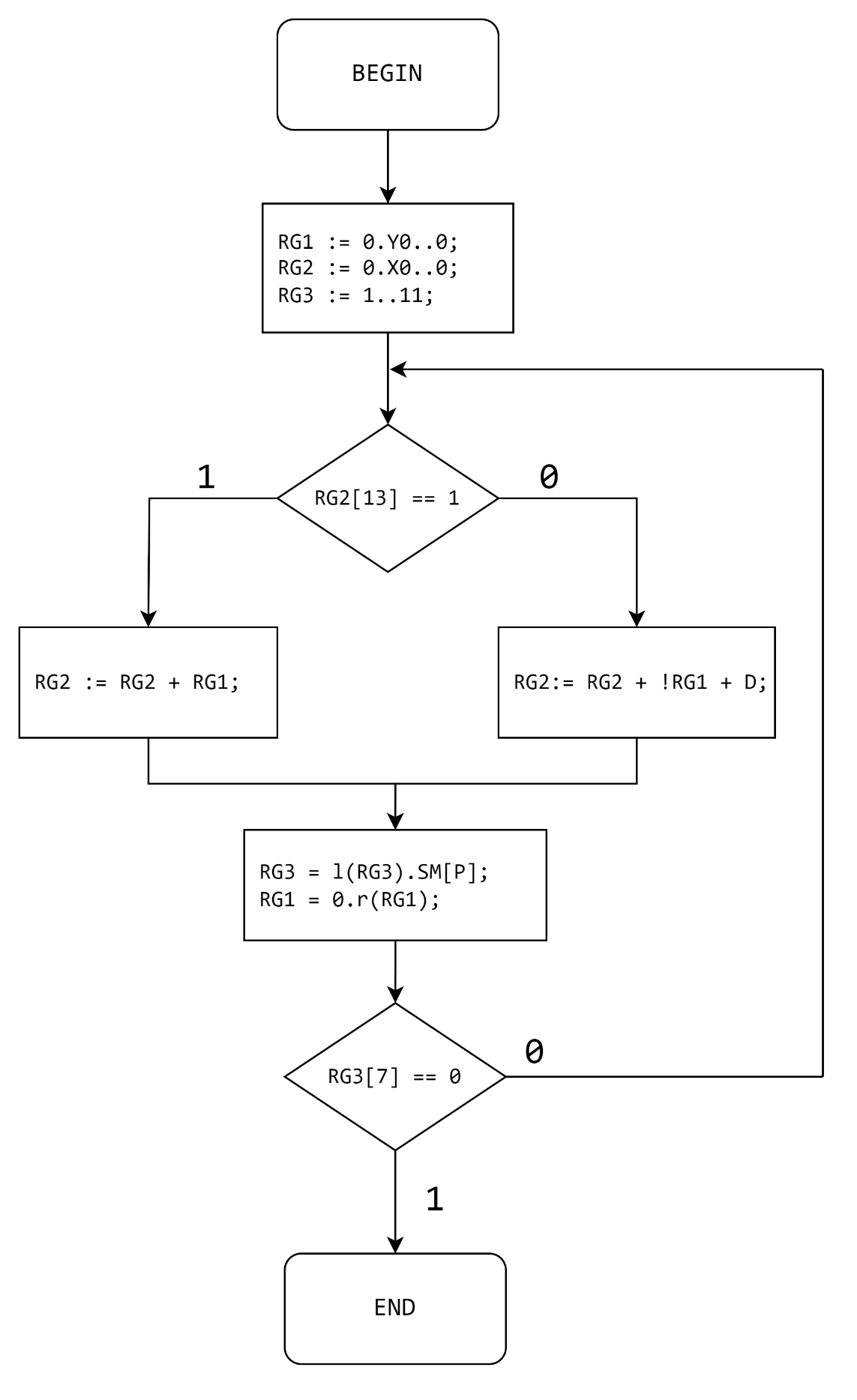


Рисунок 12 – Змістовний мікроалгоритм другого способу ділення

* + 1. *Таблиця станів регістрів 2 способу ділення*

Оскільки , то потрібно зменшити , шляхом її зсуву на один розряд вправо, тим самим додавши до одиницю. Звідси = 10001, = 010110, тоді і тепер можемо виконувати ділення

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | RG3 | RG2 | RG1 | Мікрооперації |
| - | 1111111 | **0**010110000000 | SUM: 0100001000000 SUB: 1011111000000 | RG1 = 0.Y0..0;  RG2 = 0.X0..0;  RG3 = 1..11; |
| 1 | 1111111 1111110 | 0010110000000 + 1011111000000 **1**110101000000 | SUM: 0010000100000 SUB: 1101111100000 | RG2 = RG2 – RG1;  RG1 = 0.r(RG1);  RG3 = l(RG3).SM(p) |
| 2 | 1111110 1111101 | 1110101000000 + 0010000100000 **0**000101100000 | SUM: 0001000010000 SUB: 1110111110000 | RG2 = RG2 + RG1;  RG1 = 0.r(RG1);  RG3 = l(RG3).SM(p) |
| 3 | 1111101 1111010 | 0000101100000 + 1110111110000 **1**111101010000 | SUM: 0000100001000 SUB: 1111011111000 | RG2 = RG2 – RG1;  RG1 = 0.r(RG1);  RG3 = l(RG3).SM(p) |
| 4 | 1111010 1110101 | 1111101010000 + 0000100001000 **0**000001011000 | SUM: 0000010000100 SUB: 1111101111100 | RG2 = RG2 + RG1;  RG1 = 0.r(RG1);  RG3 = l(RG3).SM(p) |
| 5 | 1110101 1101010 | 0000001011000 + 1111101111100 **1**111111010100 | SUM: 0000001000010 SUB: 1111110111110 | RG2 = RG2 – RG1;  RG1 = 0.r(RG1);  RG3 = l(RG3).SM(p) |
| 6 | 1101010 1010101 | 1111111010100 + 0000001000010 **0**000000010110 | SUM: 0000000100001 SUB: 1111111011111 | RG2 = RG2 + RG1;  RG1 = 0.r(RG1);  RG3 = l(RG3).SM(p) |
| 7 | 1010101 **0**101010 | 0000000010110 + 1111111011111 1111111110101 | SUM: 0000000010000 SUB: 1111111101111 | RG2 = RG2 – RG1;  RG1 = 0.r(RG1);  RG3 = l(RG3).SM(p) |

Результат ділення мантис: ,101010

* + 1. *Обробка порядків*

* + 1. *Форма запису нормалізованого результату*

Результат ділення мантис: ,101010

Враховуючи знаки мантис отримаємо: 0 ⊕ 1 = 1

= 1,101010 (мантиса вже є нормалізованою)

Отже, нормалізована мантиса має вигляд: 1,101010

Нормалізований результат: C = 0 | 01100 | 1 | 101010

* + 1. *Короткий формат за стандартом ANSI/IEEE 754-2008*

Число Z за стандартом ANSI/IEEE 754-2008 в короткому 32-розрядному форматі:

= 1.10001010,01010000000000000000000

* 1. *Додавання чисел (в ДК)*

Операнди: C = 0 | 01100 | 1 | 101010; Y = 0 | 00101 | 0 | 111100

* + 1. *Теоретичне обґрунтування способу виконання операцій*

Для того, щоб додати 2 числа в ДК треба:

1. Обрати максимальний порядок між Х та Y

2. На їх різницю в десятковій системі числення здвигати мантису меншого числа вправо

3. Перевести Х та Y в МПК, а потім в МДК

4. Додати в стовпчик

5. Перевести результат в ПК

6. Нормалізувати результат (за потребою)

7. Обробка порядків

8. Записати результат з плаваючою комою

* + 1. *Порівняння порядків*
    2. *Вирівнювання порядків та мантис*

Вирівняємо порядки, зсунувши мантису числа Y на 7 порядків вправо:

= 0.000101 (ПК) = 00,000000 (МПК) = 00,000000 (МДК)

* + 1. *Додавання мантис*

= 1,101010 (ПК) = 11,101010 (МПК) = 11,010110 (MДК)

= 0.000101 (ПК) = 00,000000 (МПК) = 00,000000 (МДК)

: 11,010110  
+ 00,000000  
 11,010110

Тепер переводимо результат додавання МДК в МПК шляхом інверсії всіх бітів, окрім знакових та додавання одиниці – якщо результат додавання є від’ємним числом (що в нашому випадку). Тоді:

11,010110 (МДК) = 11,101001 + 00,000001 = 11,101010 (МПК) = 1,101010 (ПК) (мантиса є вже нормалізованою)

* + 1. *Обробка порядків*

Так як , тоді порядок результуючої суми чисел є порядком ;

* + 1. *Форма запису нормалізованого результату*

Нормалізований результат: D = 0 | 01100 | 1 | 101010

* + 1. *Короткий формат за стандартом ANSI/IEEE 754-2008*

Число Z за стандартом ANSI/IEEE 754-2008 в короткому 32-розрядному форматі:

D = 1.10001010,01010000000000000000000

* 1. *Добування кореню додатного числа*

Операнди: K = |D| з додавання = 0 | 01100 | 0 | 101010

* + 1. *Теоретичне обґрунтування способу виконання операцій*

Робота з мантисою числа: один із простих алгоритмів визначення квадратного кореня з n-бітної мантиси числа базується на послідовному підборі цифр у результаті, починаючи зі старшого розряду . Для обчислення кожної i-ї цифри результату X процедура виконується таким чином. Після знаходження попередньої (i-1)-ї цифри , у i-й розряд А вставляється одиниця. Надалі обчислюється різниця . Якщо невід'ємне, приймається за правильне значення, де всі цифри розрядів відповідають цифрам результату А. У випадку, коли є від'ємним, у i-му розряді ai має бути встановлено нуль, після чого розпочинається обчислення наступного (i+1)-го розряду. Так як в цьому випадку обчислення знову починається з підстановки пробної одиниці, то замість заміни одиниці на нуль в i-му розряді віднімається одиниця з (i+1)-го розряду.

Робота з порядком числа: порядок результуючого числа удвічі менший за порядок операнда.

* + 1. *Операційна схема*

Изображение выглядит как диаграмма, Технический чертеж, План, схематичный

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 13 – Операційна схема добування кореня

* + 1. *Змістовний мікроалгоритм*

Изображение выглядит как диаграмма, текст, Технический чертеж, План

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 14 – Змістовний мікроалгоритм добування кореня

* + 1. *Таблиця станів регістрів*

Оскільки – парний, то можемо виконувати операцію:

Таблиця 7 – стани регістрів під час добування квадратного кореня з числа:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | RG1 | RG2 | RG3 |  | Мікрооперації |
| - | 000000 | **0**0000000 | 101010 | 110 | RG1 = RG2 = 0; RG3 = X;  CT = 6; |
| 1 | 000000 000001 | 00000001 00000010  + 11111111 **0**0000001 | 010100 101000 | 101 | 2(RG3 := L(RG3).0;  RG2 := L(RG2).RG3[6];)  RG2 := RG2 + !(RG1).11;  RG1 = L(RG1).(!RG2[8]);  CT = CT – 1; |
| 2 | 000001 000011 | 00000011 00000110  + 11111011 **0**0000001 | 010000 100000 | 100 | 2(RG3 := L(RG3).0;  RG2 := L(RG2).RG3[6];)  RG2 := RG2 + !(RG1).11;  RG1 = L(RG1).(!RG2[8]);  CT = CT – 1; |
| 3 | 000011  ≪000110 | 00000011 00000110  + 11110011 **1**1111001 | 000000 000000 | 011 | 2(RG3 := L(RG3).0;  RG2 := L(RG2).RG3[6];)  RG2 := RG2 + !(RG1).11;  RG1 = L(RG1).(!RG2[8]);  CT = CT – 1; |
| 4 | 000110  ≪001100 | 11110010 11100100  + 00011011 **1**1111111 | ≪000000  ≪000000 | 010 | 2(RG3 := L(RG3).0;  RG2 := L(RG2).RG3[6];)  RG2 := RG2 + RG1.11;  RG1 = L(RG1).(!RG2[8]);  CT = CT – 1; |
| 5 | 001100  ≪011001 | 11111110 11111100  + 00110011 **0**0101111 | ≪000000  ≪000000 | 001 | 2(RG3 := L(RG3).0;  RG2 := L(RG2).RG3[6];)  RG2 := RG2 + RG1.11;  RG1 = L(RG1).(!RG2[8]);  CT = CT – 1; |
| 6 | 011001  ≪110011 | 01011110 10111100  + 10011011 01010111 | ≪000000  ≪000000 | **000** | 2(RG3 := L(RG3).0;  RG2 := L(RG2).RG3[6];)  RG2 := RG2 + !(RG1).11;  RG1 = L(RG1).(!RG2[8]);  CT = CT – 1; |

Результат добування квадратного кореня мантиси: ,110011

* + 1. *Обробка порядків*
    2. *Форма запису нормалізованого результату*

Результат добування кореню: ,110011

Знак мантиси модуля – 0, тому = 0,110011 (вже є нормалізованою)

Нормалізований результат: E = 0 | 00110 | 0 | 110011

* + 1. *Короткий формат за стандартом ANSI/IEEE 754-2008*

Число E за стандартом ANSI/IEEE 754-2008 в короткому 32-розрядному форматі:

E = 0.10000100,10011000000000000000000

1. ***ПОБУДОВА ФУНКЦІОНАЛЬНИХ СХЕМ***

Оскільки, номер , то побудова схеми відбуватиметься для операції множення 3 способом, варіант парний, тому автомат Мура і , тому в базисі T-тригери.

* 1. *Функціональна схема*

Изображение выглядит как диаграмма, Технический чертеж, План, схематичный

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 15 – Функціональна схема для нормалізація результату додавання

Примітка: хоч і мантиси у нас 6 розрядні, я задля зручності та мінімізації логічних елементів, збільшив розрядність регістрів до 8, що також буде дуже зручним у подальшому можливому масштабуванні. Сигнал CH можна ототожнити із сигналом STOP, оскільки до моменту появи сигналу W значення STOP = 1, що дозволяє використовувати його як вхід CH для мультиплексора під час запису даних у регістр RG2. Після подачі сигналу W, у регістрі зберігається значення, відмінне від нуля, відповідно STOP = 0, що відповідає логіці сигналу CH — заборона повторного запису комбінації 0.X.0, оскільки вона вже була записана у RG2 раніше.

* 1. *Змістовний мікроалгоритм*

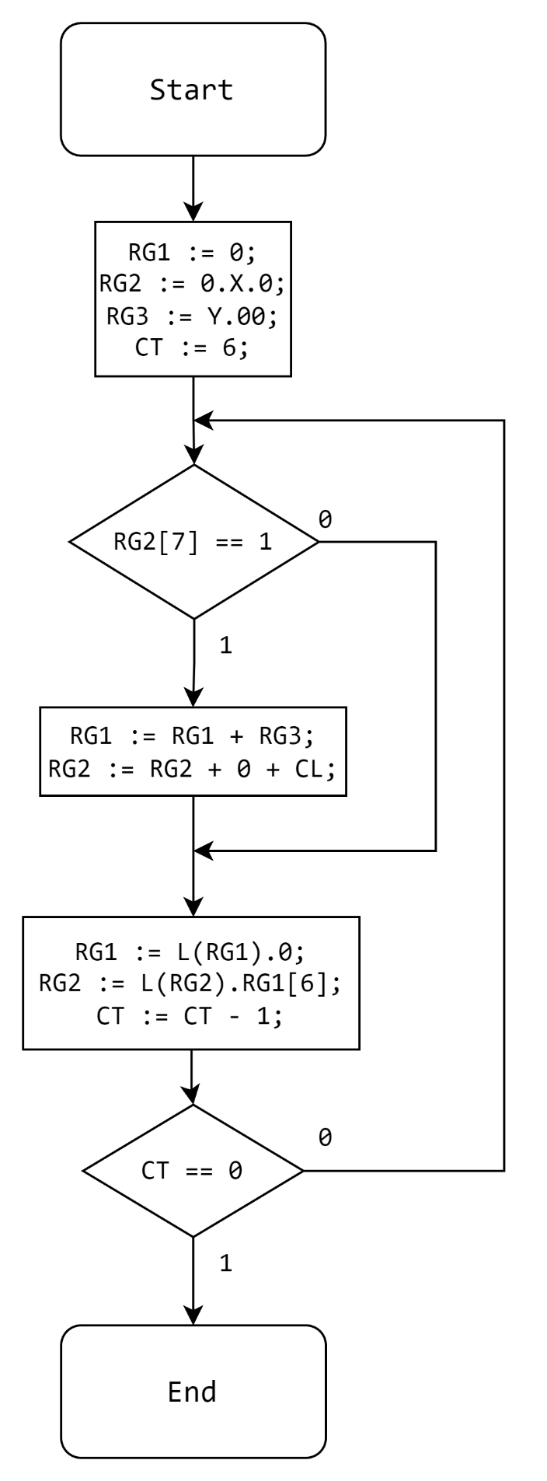


Рисунок 16 – Змістовний мікроалгоритм для нормалізація результату додавання

* 1. *Структурний мікроалгоритм*

Вивчаючи функціональну схему та подробиці мікроалгоритму, і звертаючи увагу на необхідність вставки порожнього блоку між двома послідовними сигналами, що виконуються в одному регістрі, можемо розробити наступний мікроалгоритм:

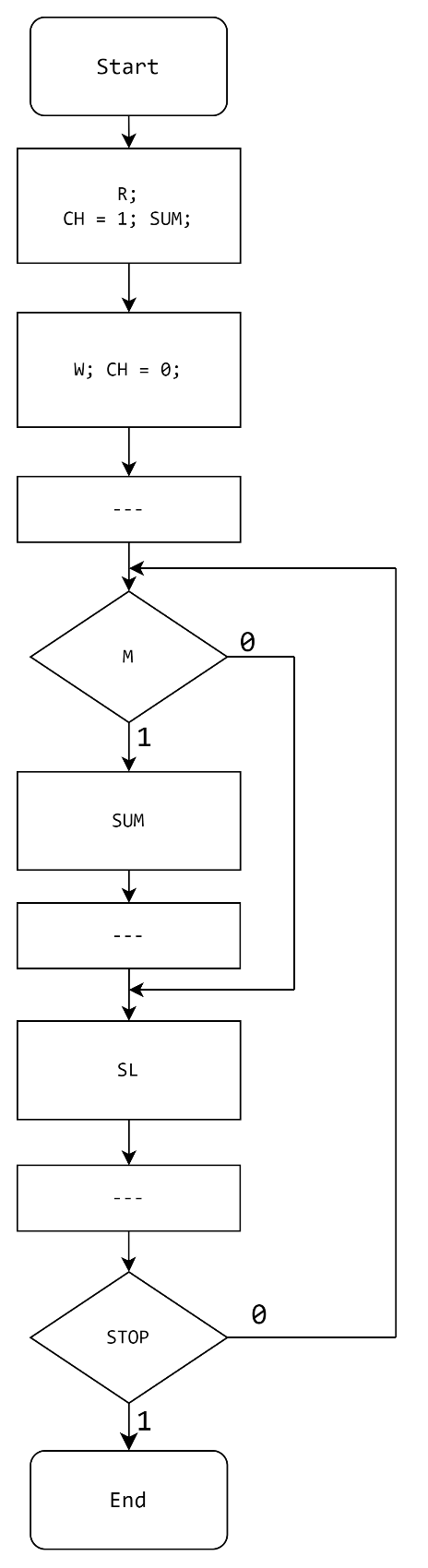


Рисунок 17 – Структурний мікроалгоритм для нормалізація результату додавання

* 1. *Автомат Мура*
     1. *Розмітка станів*

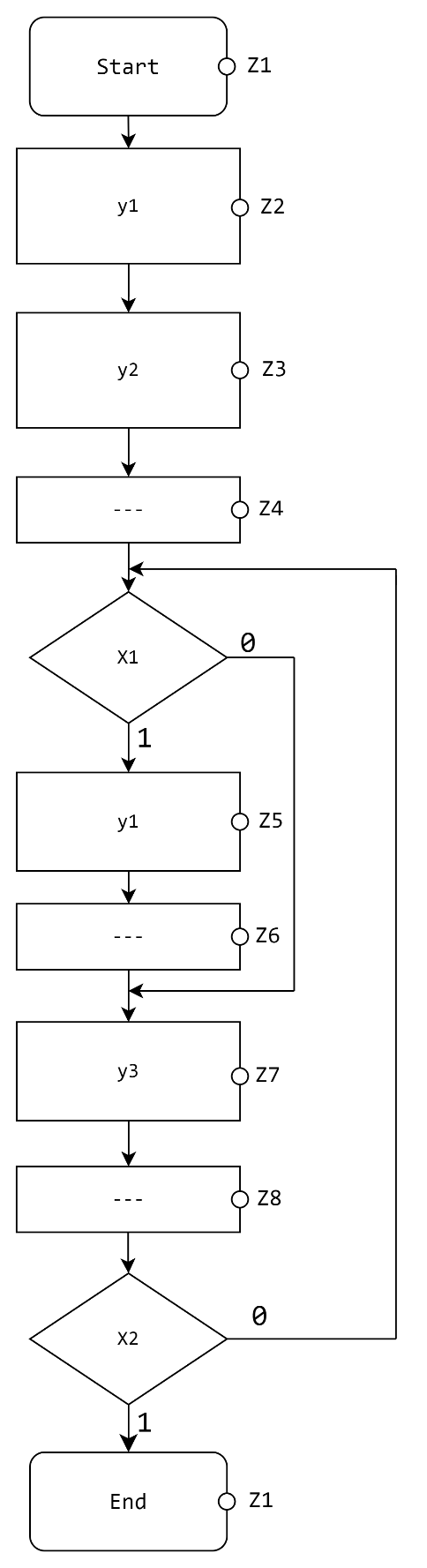


Рисунок 18 – Закодований мікроалгоритм для автомату Мура

* + 1. *Граф*

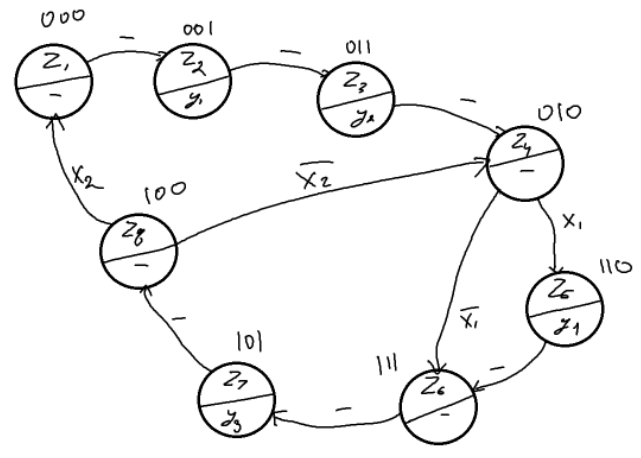


Рисунок 19 – Граф автомата Мура

Таблиця 8 – Коди станів графа

|  |  |  |  |
| --- | --- | --- | --- |
| Стан | Q3 | Q2 | Q1 |
| Z1 | 0 | 0 | 0 |
| Z2 | 0 | 0 | 1 |
| Z3 | 0 | 1 | 1 |
| Z4 | 0 | 1 | 0 |
| Z5 | 1 | 1 | 0 |
| Z6 | 1 | 1 | 1 |
| Z7 | 1 | 0 | 1 |
| Z8 | 1 | 0 | 0 |

* + 1. *Структурна таблиця*

Таблиця 9 – Структурна таблиця

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ПС | Код ПС | | | СП | Код СП | | | Логічна  умова | | Керуючі  сигнали | | | Функції збудження тригерів | | |
|  |
|  | | |  | | |  |  |  |  |  |  |  |  |  | |
| Z1 | 0 | 0 | 0 | Z2 | 0 | 0 | 1 | - | - | 0 | 0 | 0 | 0 | 0 | 1 |  | |
| Z2 | 0 | 0 | 1 | Z3 | 0 | 1 | 1 | - | - | 1 | 0 | 0 | 0 | 1 | 0 |  | |
| Z3 | 0 | 1 | 1 | Z4 | 0 | 1 | 0 | - | - | 0 | 1 | 0 | 0 | 0 | 1 |  | |
| Z4 | 0 | 1 | 0 | Z5 | 1 | 1 | 0 | 1 | - | 0 | 0 | 0 | 1 | 0 | 0 |  | |
| Z4 | 0 | 1 | 0 | Z6 | 1 | 1 | 1 | 0 | - | 0 | 0 | 0 | 1 | 0 | 1 |  | |
| Z5 | 1 | 1 | 0 | Z6 | 1 | 1 | 1 | - | - | 1 | 0 | 0 | 0 | 0 | 1 |  | |
| Z6 | 1 | 1 | 1 | Z7 | 1 | 0 | 1 | - | - | 0 | 0 | 0 | 0 | 1 | 0 |  | |
| Z7 | 1 | 0 | 1 | Z8 | 1 | 0 | 0 | - | - | 0 | 0 | 1 | 0 | 0 | 1 |  | |
| Z8 | 1 | 0 | 0 | Z4 | 0 | 1 | 0 | - | 0 | 0 | 0 | 0 | 1 | 1 | 0 |  | |
| Z8 | 1 | 0 | 0 | Z1 | 0 | 0 | 0 | - | 1 | 0 | 0 | 0 | 1 | 0 | 0 |  | |

* + 1. *Одержання мінімізованих функцій*

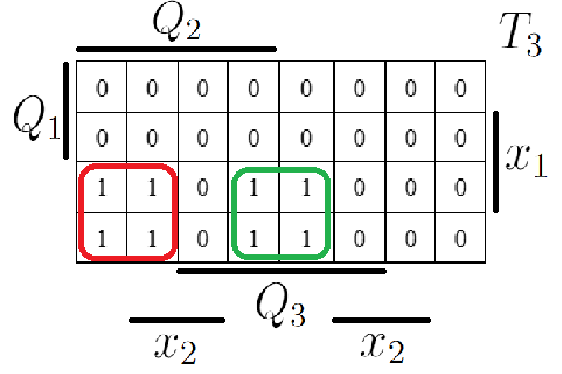
Оскільки в аналізі використовується булевий базис, існує можливість мінімізувати вирази через диз'юнктивну нормальну форму (ДДНФ) або кон'юнктивну нормальну форму (ДКНФ). В даному контексті оптимальніше застосовувати ДДНФ. Тому для мінімізації функцій збудження тригерів ми використаємо таблиці Вейча, де мінімізовані комбінації позначені однаковим кольором.

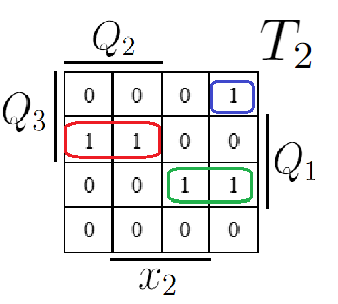
Функції, які потрібно мінімізувати:

(мінімізацію провести неможливо)

(мінімізацію провести неможливо)

(мінімізацію провести неможливо)





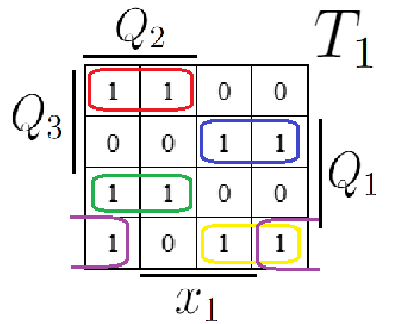


Рисунок 20-22 – Мінімізація таблицями Вейча функцій збудження тригерів

Результат МДНФ в елементному базисі 4І/4АБО (операторна форма):

* + 1. *AFDK та часові діаграми*

Изображение выглядит как диаграмма, План, Технический чертеж, схематичный

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 23 – Пристрій обчислення добутку із керуючим автоматом за моїм варіантом

Изображение выглядит как снимок экрана, дисплей, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 24 – Часова діаграма пристрою разом із керуючим автоматом

Приклад роботи схеми для інших операндів:

Изображение выглядит как диаграмма, План, текст, Технический чертеж

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 25 - Пристрій обчислення із керуючим автоматом для довільних операндів

Изображение выглядит как снимок экрана, дисплей, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 26 – Часова діаграма пристрою разом із керуючим автоматом

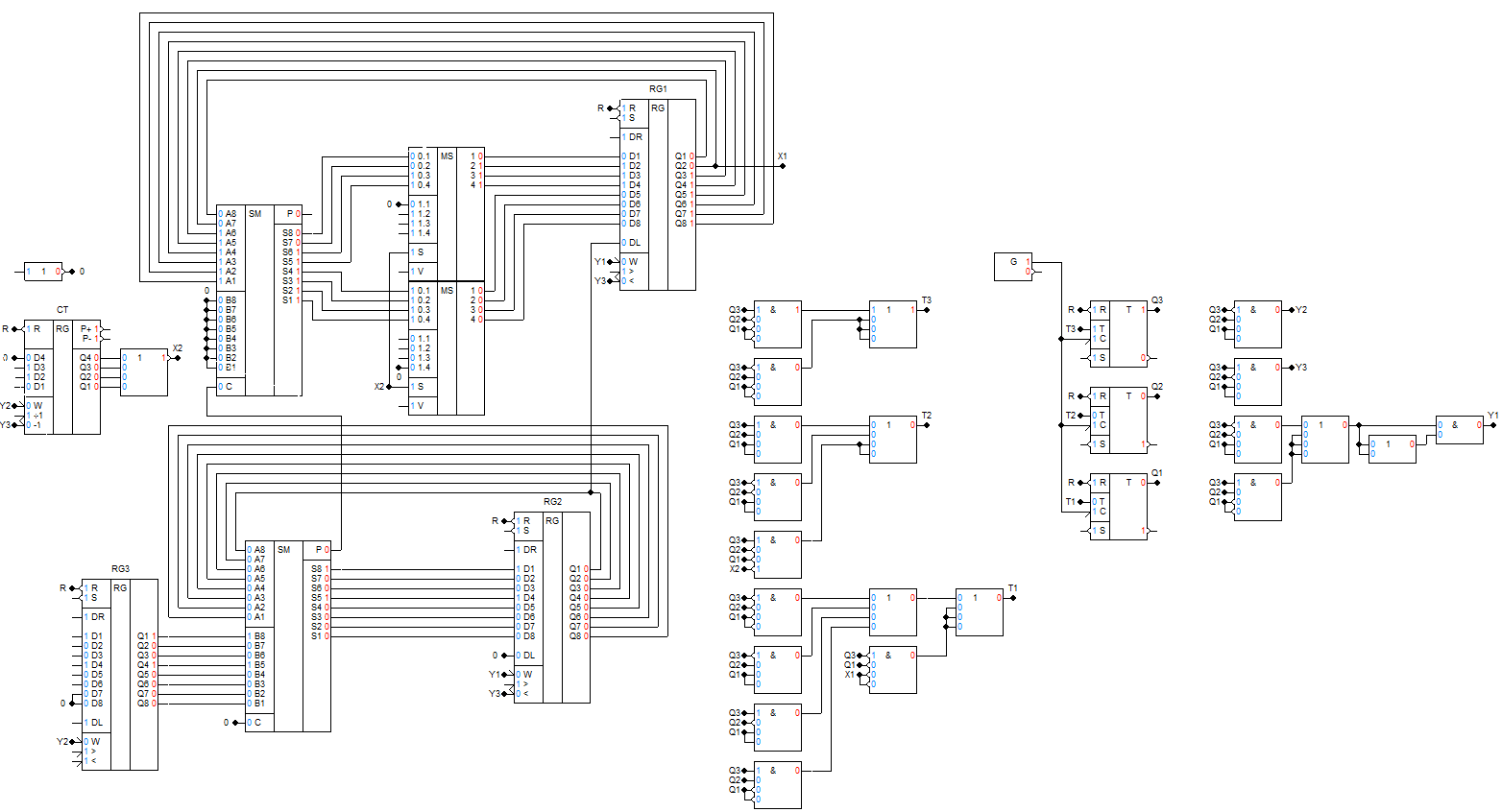


Рисунок 27 - Пристрій обчислення із керуючим автоматом для довільних операндів

Изображение выглядит как снимок экрана, дисплей, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 28 – Часова діаграма пристрою разом із керуючим автоматом

1. ***ВИСНОВОК***

У ході курсової роботи було виконано аналіз та теоретичні розрахунки для різних арифметичних операцій: множення, ділення, додавання та добування арифметичного квадратного кореня. Розроблено структурний автомат, який реалізує метод множення третім способом. Детальний опис функціональної електричної схеми цього автомата представлено в розділі "Автомат керуючий. Схема електрична функціональна", оформленому згідно з вимогами Єдиної системи конструкторської документації (ЄСКД). Результати дослідження підтвердили високу ефективність роботи автомата, що свідчить про його придатність до використання в обчислювальній техніці. Проект сприяв глибшому розумінню теоретичних основ арифметичних операцій, удосконаленню практичних навичок у розробці схем з керуючим автоматом, а також формуванню вмінь працювати з технічною документацією та електронним обладнанням.

1. ***СПИСОК ЛІТЕРАТУРИ***

1. Жабін В.І., Верба О.А. Комп’ютерна логіка. Курсова робота. Навчальний посібник. https://ela.kpi.ua/handle/123456789/50134

2. Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В. Прикладна теорія цифрових автоматів: Навч. посібник.– К.: Вид-во НАУ, 2009.– 364 с.

3. Жабін В.І., Ткаченко В.В. Цифрові автомати. Практикум. – К.: ВЕК+, 2004.– 160 с.

4. ДСТУ 3008-2015 «Державний стандарт України. Документація. Звіти в сфері науки і техніки. Структура і правила оформлення».

5. Матвієнко М.П. Комп’ютерна логіка. Підручник. Вид. 2-ге перероб. та доп. – Київ: Видавництво Ліра – К, 2017. – 324 с.

6. ДСТУ 3008-2015 «Державний стандарт України. Документація. Звіти в сфері науки і техніки. Структура і правила оформлення».

7. ДСТУ ISO 5457:2006 (ISO 5457:1999, IDТ) Національний стандарт України. Документація технічна на вироби. Кресленики. Розміри та формати.

8. ДСТУ ГОСТ 2.702:2013 ЄСКД. Правила виконання електричних схем (ГОСТ 2.702-2011, IDT).

9. ДСТ 2.104-2006. ЄСКД. Основні написи (Міждержавний. В Україні -ДСТУ 2.104-2006).