НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського"

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра обчислювальної техніки

РОЗРАХУНКОВА ГРАФІЧНА РОБОТА

з дисципліни „Комп'ютерна логіка 2. Комп'ютерна арифметика”

Виконала

Молчанова В. С.

Група ІВ-71

Факультет ІОТ

Залікова книжка № 7110

Керівник: Жабін В. І.

Київ – 2018 р.

#### Завдання

1. Числа  і  в прямому коді записати у формі з плаваючою комою у класичному варіанті (з незміщеним порядком і повною мантисою). На порядок відвести 4 розряди, на мантису 7 розрядів (з урахуванням знакових розрядів). Записати числа і  також за стандартом ANSI/IEEE 754-2008 в короткому 32-розрядному форматі).

2. 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). Номери операцій (для п.3) відповідають порядку переліку, починаючи з нуля (наприклад, 0 – множення першим способом; 5 – ділення другим способом). Операндами для першого способу множення є задані числа та . Для кожної наступної операції першим операндом є результат попередньої операції, а другим операндом завжди є число . (Наприклад, для ділення першим способом першим операндом є результат множення за четвертим способом, для операції обчислення кореня операндом є результат додавання зі знаком плюс).

Для обробки мантис кожної операції, подати:

2.1 теоретичне обґрунтування способу;

2.2 операційну схему;

2.3 змістовний (функціональний) мікроалгоритм;

2.4 таблицю станів регістрів (лічильника), довжина яких забезпечує одержання 6 основних розрядів мантиси результату;

2.5 обробку порядків (показати у довільній формі);

2.6 форму запису нормалізованого результату з плаваючою комою в пам’ять комп’ютера в прямому коді.

Вказані пункти для операції додавання виконати для етапу нормалізації результату з урахуванням можливого нулевого результату. Інші дії до етапу нормалізації результату можна проілюструвати у довільній формі.

3 Для операції з номером  додатково виконати:

3.1 побудувати функціональну схему з відображенням управляючих сигналів, входів для запису операндів при ініціалізації пристрою і схем формування внутрішніх логічних умов;

3.2 розробити закодований (структурний) мікроалгоритм (мікрооперації замінюються управляючими сигналами виду W,SL,SR тощо);

3.3 для операції з парним двійковим номером  додатково подати граф управляючого автомата Мура з кодами вершин, а для непарного номера – автомата Мілі;

3.4 побудувати управляючий автомат на тригерах та елементах булевого базису. Вибрати -тригери для автомата Мура та -тригери для автомата Мілі.

#### Визначення та обґрунтування варіанту:

Перевести номер залікової книжки в двійкову систему. Записати два 10-розрядних двійкових числа:

 і ,

де  - двійкові цифри номера залікової книжки у двійковій системі числення ( - молодший розряд).

711010=1110001002;

 = -101000,1100

 = +111100,0100

#### Основна частина

### Завдання №1

= 1.101000,1100; = 0.111100,0100;

Представлення чисел у формі з плаваючою точкою з порядком і мантисою:

X2:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 1 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |

Y2:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 1 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |

Представлення чисел за стандартом ANSI/IEEE 754-2008 в короткому 32-розрядному форматі

Ex = 127 + 6 = 13310 = 100001012

X2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Порядок

Мантиса

Ey = 127 + 6 = 13310 = 100001012

Y2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Порядок

Мантиса

### Завдання №2

2.1 Перший спосіб множення**.**

# 2.1.1 Теоретичне обґрунтування першого способу множення:

Числа множаться у прямих кодах, знакові та основні розряди обробляються окремо. Для визначення знака добутку здійснюють підсумування по модулю 2 цифр, що розміщуються в знакових розрядах співмножників.

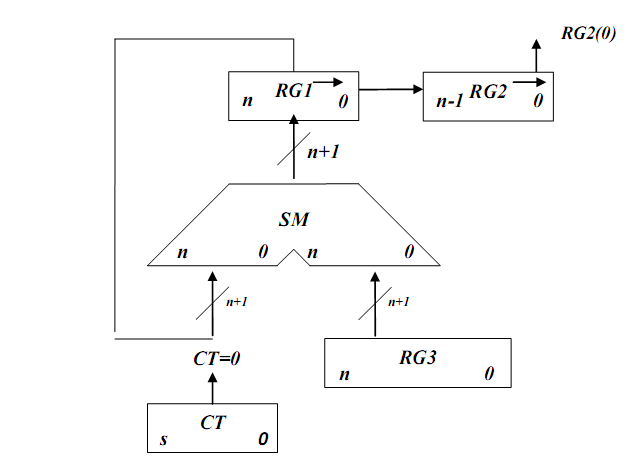
Множення мантис першим способом здійснюється з молодших розрядів множника, сума часткових добутків зсувається вправо, а множене залишається нерухомим. Тоді добуток двох чисел представляється у вигляді:

Z=YХ=+ Y…+ Y =

= ((..((0+Y)+ Y)+…+ Y) +…+ Y);

Z = ;

2.1.2 Операційна схема**:**



*Рисунок 2.1.1- Операційна схема*

# 2.1.3 Змістовний мікроалгоритм:

Початок

RG1:=0; RG2:=X; RG3:=Y; CT:=6;

RG2[0]

RG1:=RG1+RG3;

RG1:=0.r(RG1); RG2:=RG1(n).r(RG2); CT:=CT-1;

CT=0

Кінець

1

0

1

0

*Рисунок 2.1.2 - Змістовний мікроалгоритм виконання операції множення першим способом*

# 2.1.4 Таблиця станів регістрів:

*Таблиця 2.1.1 -Таблиця станів регістрів*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **RG1** | **RG2** | **RG3** | **CT** |
| **пс** | 0 | 101000 | 111100 | 110 |
| **1** | 0 | 010100 |  | 101 |
| **2** | 0 | 001010 |  | 100 |
| **3** | 0 | 000101 |  | 011 |
| **4** | 111100  011110 | 000010 |  | 010 |
| **5** | 001111 | 000001 |  | 001 |
| **6** | 111100  1001011  **100101** | **100000** |  | 000 |

# 2.1.5 Обробка порядків:

Порядок добутку буде дорівнювати сумі порядків множників з урахуванням знаку порядків:

=6; =6; =1210=11002

2.1.6 Нормалізація результату**:**

Отримали результат: 100101100000

Знак мантиси: 1 0 = 1.

Нормалізація не потрібна.

Запишемо нормалізований результат:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

## 2.2 Другий спосіб множення.

# 2.2.1 Теоретичне обґрунтування другого способу множення:

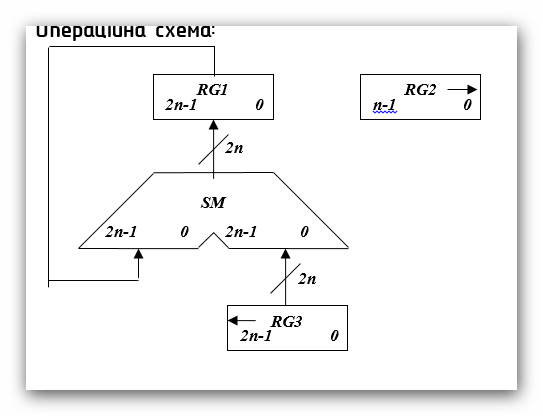
Числа множаться у прямих кодах, знакові та основні розряди обробляються окремо. Визначення знака добутку здійснюють підсумування по модулю 2 цифр, що розміщуються в знакових розрядах співмножників.

Множення мантис другим способом здійснюється з молодших розрядів, множене зсувається вліво, а сума часткових добутків залишається нерухомою.

Z=Y+ Y…+ Y; Z=((0+ Y)+ Y)…+ Y;

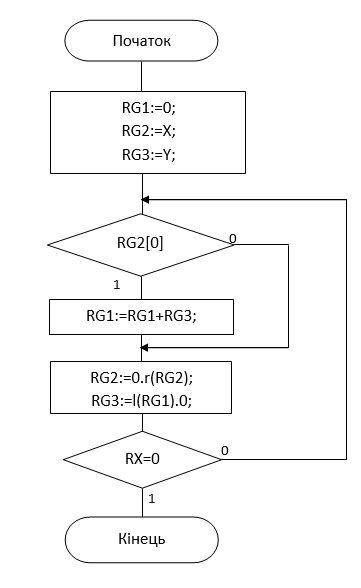
Z=;

# 2.2.2 Операційна схема:

****

*Рисунок 2.2.1 - Операційна схема*

# 2.2.3 Змістовний мікроалгоритм:



*Рисунок 2.2.2 - Змістовний мікроалгоритм.*

# 2.2.4 Таблиця станів регістрів:

*Таблиця 2.2.1 -Таблиця станів регістрів*

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **RG1** | **RG3 ←** | **RG2 →** |
| **ПС** | 0 | 000000111100 | 100101 |
| **1** | 000000111100 | 000001111000 | 010010 |
| **2** | 000000111100 | 000011110000 | 001001 |
| **3** | 000011110000  000100101100 | 000111100000 | 000100 |
| **4** | 000100101100 | 001111000000 | 000010 |
| **5** | 000100101100 | 011110000000 | 000001 |
| **6** | 011110000000  **100010101100** | 111100000000 | 000000 |

# 2.2.5 Обробка порядків:

Порядок добутку буде дорівнювати сумі порядків множників з урахуванням знаку порядків:

=12; =6; =1810=100102

# 2.2.6 Нормалізація результату:

Отримали результат: 100010101100

Знак мантиси: 1 0 = 1.

Нормалізація мантиси не потрібна.

Запишемо нормалізований результат:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

## 2.3 Третій спосіб множення.

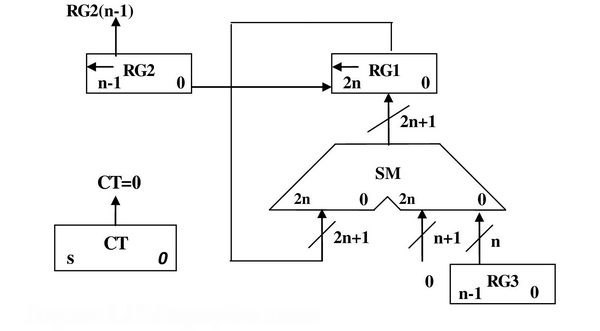
# 2.3.1 Теоретичне обгрунтування третього способу множення:

Числа множаться у прямих кодах, знакові та основні розряди обробляються окремо. Визначення знака добутку здійснюють підсумування по модулю 2 цифр, що розміщуються в знакових розрядах співмножників.

Множення мантис третім способом здійснюється зі старших розрядів множника, сума часткових добутків і множник зсуваються вліво, а множене нерухоме.

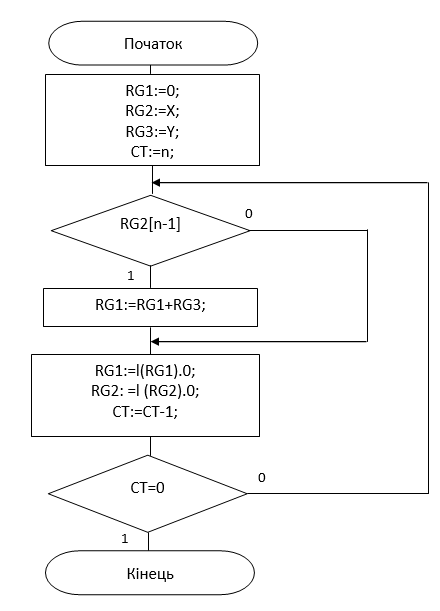
Z=Y+ Y…+ Y; ; Z=Y+2(Y+2(Y…+2Y)); Z=;

# 2.3.2 Операційна схема:



*Рисунок 2.3.1 - Операційна схема*

# 2.3.3 Змістовний мікроалгоритм:



*Рисунок 2.3.2* **-** *Змістовний мікроалгоритм*

# 2.3.4 Таблиця станів регістрів:

*Таблиця 2.3.1- Таблиця станів регістрів*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **RG1 ←** | **RG2 ←** | **RG3** | **CT** |
| **пс** | 0 | 100010 | 111100 | 110 |
| **1** | 0000001111000 | 000100 |  | 101 |
| **2** | 0000011110000 | 001000 |  | 100 |
| **3** | 0000111100000 | 010000 |  | 011 |
| **4** | 0001111000000 | 100000 |  | 010 |
| **5** | 0000000111100  0001111111100  0011111111000 | 000000 |  | 001 |
| **6** | **0111111110000** | 000000 |  | 000 |

# 2.3.5 Обробка порядків:

Порядок добутку буде дорівнювати сумі порядків множників з урахуванням знаку порядків:

=18; =6; =2410=110002

# 2.3.6 Нормалізація результату:

Отримали результат: 0111111110000

Знак мантиси: 1 0 = 1.

Робимо зсув результату вліво, доки у першому розряді не буде одиниця,

Порядок зменшуємо на 1:

11111111000; =232=101112;

Запишемо нормалізований результат:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

2.4 Четвертий спосіб множення**.**

# 2.4.1 Теоритичне обґрунтування четвертого способу множення:

Числа множаться у прямих кодах, знакові та основні розряди обробляються окремо. Визначення знака добутку здійснюють підсумування по модулю 2 цифр, що розміщуються в знакових розрядах співмножників.

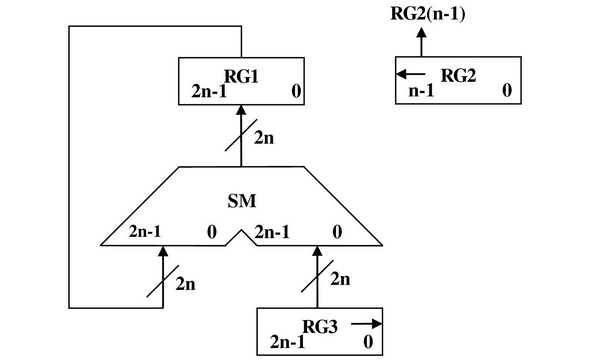
Множення здійснюється зі старших розрядів множника, сума часткових добутків залишається нерухомою, множене зсувається праворуч, множник ліворуч.

.

*.*

з початковими значеннями i=1, Y0=2-1Y, Z0=0.

# 2.4.2 Операційна схема:



*Рисунок 2.4.1-* *Операційна схема*

# 2.4.3 Змістовний мікроалгоритм:

RG1:=0;

RG2:=X;

RG3:=Y;

RG3:=0.r(RG3)

RG2[n-1]

RG1:=RG1+RG3;

RG3:=0.r(RG3)

RG2:=l(RG2).0

RG2=0

Початок

Кінець

1

1

0

0

*Рисунок 2.4.2* **-** *Змістовний мікроалгоритм*

# 2.4.4 Таблиця станів регістрів:

*Таблиця 2.4.1- Таблиця станів регістрів*

|  |  |  |  |
| --- | --- | --- | --- |
| № | RG1 | RG3 → | RG2 ← |
| ПС | 0 | 011110000000 | 111111 |
| **1** | 011110000000 | 001111000000 | 111110 |
| **2** | 001111000000  101101000000 | 000111100000 | 111100 |
| **3** | 000111100000  110100100000 | 000011110000 | 111000 |
| **4** | 000011110000  111000010000 | 000001111000 | 110000 |
| **5** | 000001111000  111010001000 | 000000111100 | 100000 |
| **6** | 000000111100  **111011000100** | 000000011110 | 000000 |

# 2.4.5 Обробка порядків:

Порядок добутку буде дорівнювати сумі порядків множників з урахуванням знаку порядків:

=23; =6; =2910=111012

# 2.4.6 Нормалізація результату:

Отримали результат: 101000100000

Знак мантиси: 1 0 = 1.

Нормалізація мантиси не потрібна.

Запишемо нормалізований результат:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

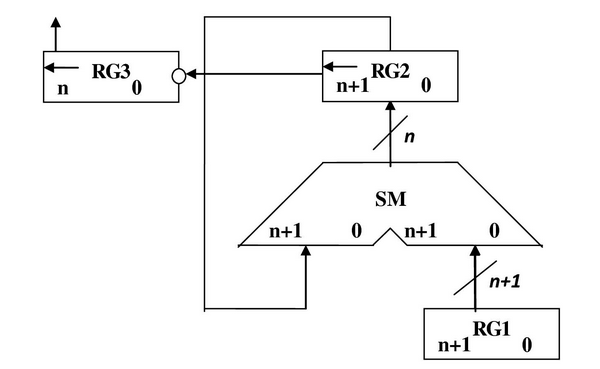
## 2.5. Першиий спосіб ділення.

# 2.5.1 Теоритичне обґрунтування першого способу ділення:

Нехай ділене Х і дільник Y є n-розрядними правильними дробами, поданими в прямому коді. В цьому випадку знакові й основні розряди операндів обробляються окремо. Знак результату визначається шляхом підсумовування по модулю 2 цифр, записаних в знакових розрядах.

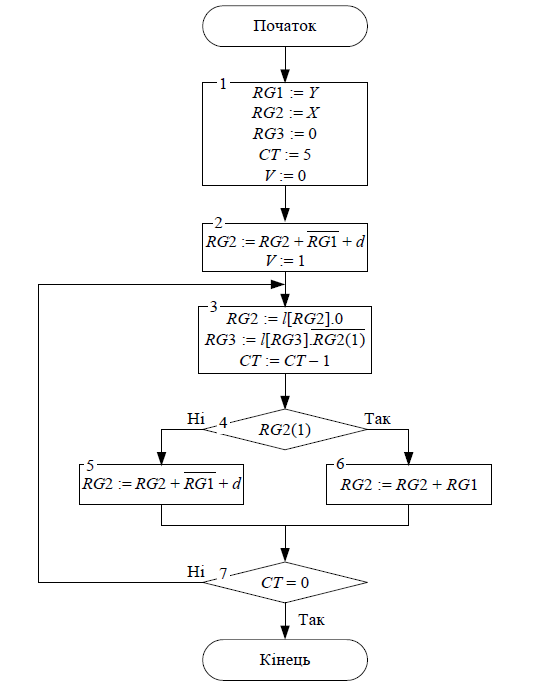
При реалізації ділення за першим методом здійснюється зсув вліво залишку при нерухомому дільнику. Черговий залишок формується в регістрі RG2 (у вихідному стані в цьому регістрі записаний Х). Виходи RG2 підключені до входів СМ безпосередньо, тобто ланцюги видачі коду з RG2 не потрібні. Час для підключення n+1 цифри частки визначається виразом t=(n+1)(tt+tc), де tt - тривалість виконання мікрооперації додавання-віднімання; tc - тривалість виконання мікрооперації зсуву.

# 2.5.2 Операційна схема:



*Рисунок 2.5.1-* *Операційна схема*

# 2.5.3 Змістовний мікроалгоритм:



*Рисунок 2.5.2-Змістовний мікроалгоритм*

# 2.5.4 Таблиця станів регістрів:

|  |  |  |  |
| --- | --- | --- | --- |
| *Таблиця 2.5.1- Таблиця станів регістрів* | | | |
| **№** | **RG3(Z)** | **RG2(X)** | **RG1(Y)** |
| **пс** | 000000 | 00111011  11000100  11111111 | 00111100  11000100 |
| **1** | 000000 | 11111110  00111100  00111010 |  |
| **2** | 000001 | 01110100  11000100  00111000 |  |
| **3** | 000011 | 01110000  11000100  00110100 |  |
| **4** | 000111 | 01101000  11000100  00101100 |  |
| **5** | 001111 | 01011000  11000100  00011100 |  |
| **6** | 011111 | 00111000  11000100  11111100 |  |
| **7** | **111110** | 11111000  00111100  00110100 |  |

# 2.5.5 Обробка порядків:

Порядок частки буде дорівнювати:

=29; =6; =2310=101112;

# 2.5.6 Нормалізація результату:

Отримали результат: 11111011111

Знак мантиси: 1 0 = 1.

Нормалізація мантиси не потрібна.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

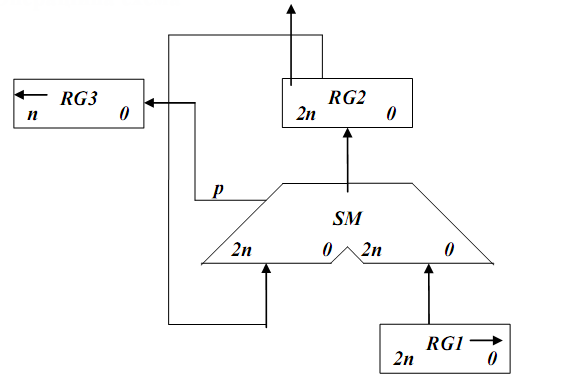
## 2.6. Другий спосіб ділення.

# 2.6.1 Теоритичне обгрунтування другого способу ділення:

Нехай ділене Х і дільник Y є n-розрядними правильними дробами, поданими в прямому коді. В цьому випадку знакові й основні розряди операндів обробляються окремо. Знак результату визначається шляхом підсумовування по модулю 2 цифр, записаних в знакових розрядах.

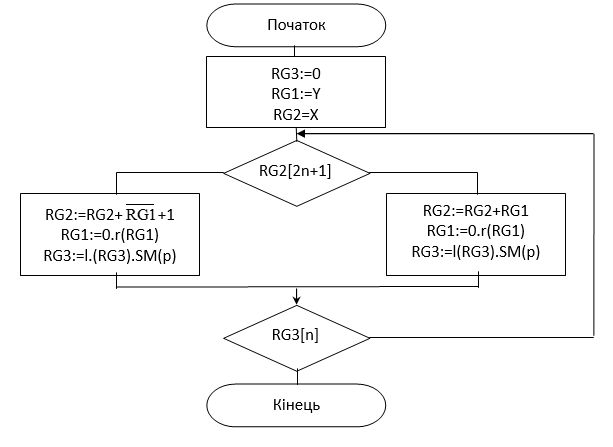
Остача нерухома, дільник зсувається праворуч. Як і при множенні з нерухомою сумою часткових добутків можна водночас виконувати підсумування і віднімання, зсув в регістрах Y,Z. Тобто 1 цикл може складатися з 1 такту, це дає прискорення відносно 1-го способу.

# 2.6.2 Операційна схема



*Рисунок 2.6.1-Операційна схема*

# 2.6.3 Змістовний мікроалгоритм



*Рисунок 2.6.2-Змістовний мікроалгорит*

# 2.6.4 Таблиця станів регістрів

*Таблиця 2.6.1- Таблиця станів регістрів*

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **RG3(Z)** | **RG2(X)** | **RG1(Y)** |
| **ПС** | 1111111 | 0111110000000 | 0111100000000 |
| **1** | 1111111\* | 1000100000000  0000010000000 | 0011110000000 |
| **2** | 111111\*0 | 1100010000000  1100100000000 | 0001111000000 |
| **3** | 11111\*00 | 0001111000000  1110011000000 | 0000111100000 |
| **4** | 1111\*000 | 0000111100000  1111010100000 | 0000011110000 |
| **5** | 111\*0000 | 0000011110000  1111110010000 | 0000001111000 |
| **6** | 11\*00001 | 0000001111000  0000000001000 | 0000000111100 |
|  | **1\*000010** | 1111111000100  1111111001100 | 0000000011110 |

# 2.6.5 Обробка порядків:

Порядок частки буде дорівнювати:

В моєму випадку =23; =6; =1710=100012;

# 2.6.6 Нормалізація результату:

Отримали результат: 1.000010

Знак мантиси: 1 0 = 1.

Робимо зсув вправо, доки молодший розряд цілої часини не стане 0

Порядок збільшуємо на 1:

0.100001; =1810=100102

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

## 2.7. Операція додавання чисел.

# 2.7.1 Теоретичне обґрунтування способу

В пам’яті числа зберігаються у ПК. На першому етапі додавання чисел з плаваючою комою виконують вирівнювання порядків до числа із старшим порядком. На другому етапі виконують додавання мантис. Додавання мантис виконується у доповнювальних кодах, при необхідності числа у ДК переводяться в АЛП. Додавання виконується порозрядно на n-розрядному суматорі з переносом. Останній етап – нормалізація результату. Виконується за допомогою зсуву мантиси результату і коригування порядку результату. Порушення нормалізації можливо вліво і вправо, на 1 розряд вліво і на n розрядів вправо.

# 2.7.2 Порівняння порядків.

PX = 0.10010 MX = 1.100001

PY = 0.00110 MY = 0.111100

1810-610=1210=11002

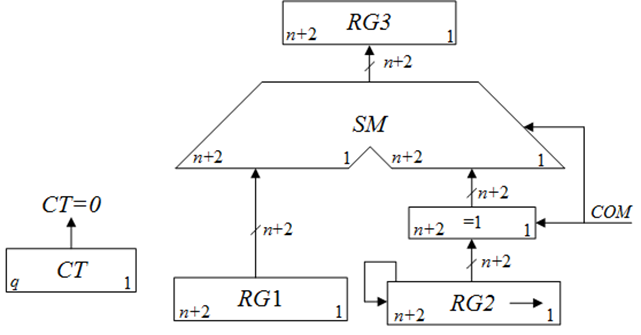
min(ΔP, n) = 610 = 1102

Числа у МДК:

X = 11.011111

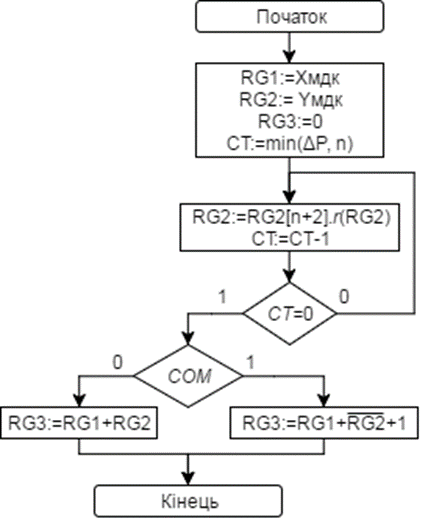
Y = 00.010000

# 2.7.3 Операційна схема



*Рисунок 4.1. Операційна схема пристрою додавання/віднімання у МДК*

# 2.7.4 Змістовний мікроалгоритм



*Рисунок 4.2. Змістовний мікроалгоритм пристрою додавання/віднімання у МДК*

*Таблиця 2.7.1. Таблиця станів регістрів пристрою додавання/віднімання у МДК двох чисел*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ ц.** | **RG1** | **RG2🡪** | **RG3** | **CT** |
| П.С. | 11011111 | 00010000 | 00000000 | 110 |
| 1 | 11011111 | 00001000 | 00000000 | 101 |
| 2 | 11011111 | 00000100 | 00000000 | 100 |
| 3 | 11011111 | 00000010 | 00000000 | 011 |
| 4 | 11011111 | 00000001 | 00000000 | 010 |
| 5 | 11011111 | 00000000 | 00000000 | 001 |
| 6 | 11011111 | 00000000 | 00000000 | 000 |
| - | 11001101 | 00000000 | 11011111  +00000000  **11011111ДК**  **00100001ПК** | 000 |

**2.7.5 Обробка порядків**

Мантиса: Mz = 1.100001; Pz = 100102;

Знак мантиси: 1.

**2.7.6 Нормалізація результату**

Результат додавання Z=X+Y.

Отримали результат: 100001

Знак мантиси: 1 0 = 1.

В даному випадку нормалізація не потрібна.

Mz = 0.100001; =1810=100102;

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

## 2.8.Операція добування кореня

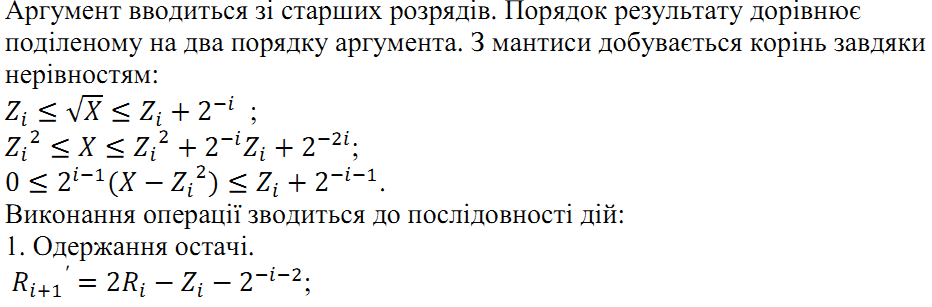
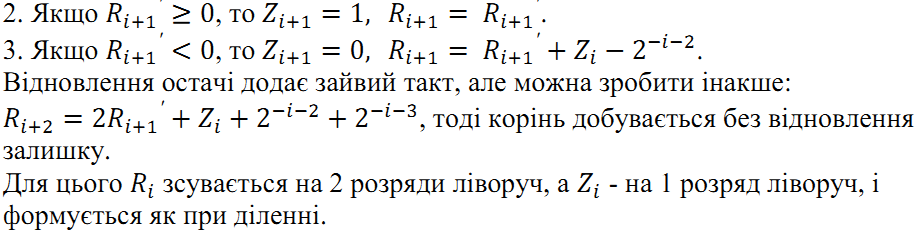
# 2.8.1 Теоритичне обґрунтування операції обчислення квадратного кореня

Аргумент вводиться зі старших розрядів. Порядок результату дорівнює поділеному на 2 порядку аргумента. З мантиси добувається корінь завдяки нерівностям:

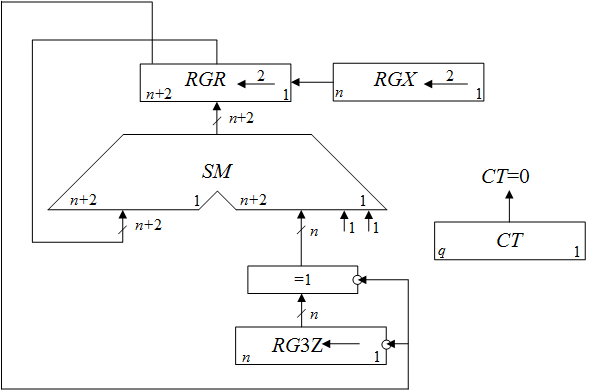
; ;

.

Виконання операції зводиться до послідовності дій:

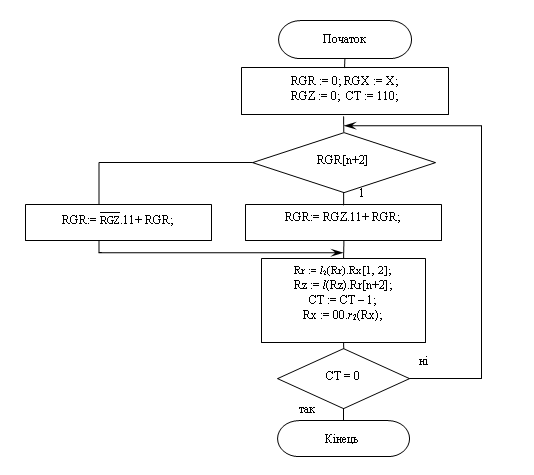


2.8.2 Операційна схема операції обчислення квадратного кореня

****

*Рисунок 2.8.1 –Операційна схема*

# 2.8.3 Змістовний мікроалгоритм

**

*Рисунок 2.8.2 – Змістовний мікроалгоритм*

# 2.8.4 Таблиця станів регістрів

*Таблиця 2.8.1 – Таблиця станів регістрів*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | RGZ < | RGR << | RGX << | СТ |
| **ПС** | 000000 | 00000000  **0**0000010 | 100001  000100 | 110 |
| **ПЗ** |
| **1** | 000001 | +11111111  00000001  **0**0000100 | 010000 | 101 |
| **2** | 000010 | +11111011  11111111  **1**1111101 | 000000 | 100 |
| **3** | 000101 | +00001011  00001000  **0**0100000 | 000000 | 011 |
| **4** | 001011 | +11101011  00001011  **0**0101100 | 000000 | 010 |
| **5** | 010110 | +11010011  11111111  **1**1111100 | 000000 | 001 |
| **6** | **101101** | +01011011  00010111  **0**1011100 | 000000 | 000 |

**2.8.8 Обробка порядків**

В моєму випадку =9;

**2.8.9 Запис результату**

Отримали результат Z = 101101;

Результат нормалізований, готовий до запису у мантису:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |

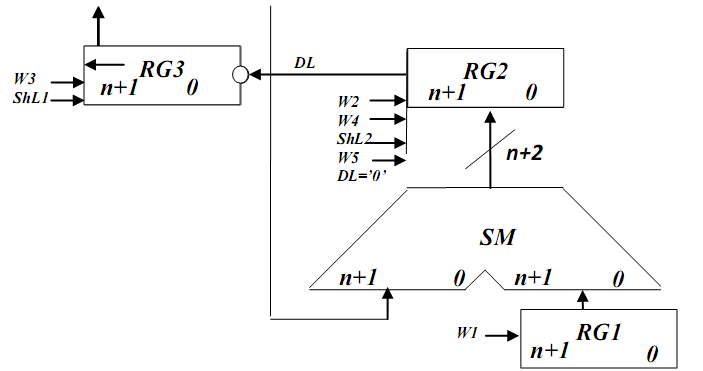
### Завдання 3

x3x2x1 = 1002 – парне.

Синтез управляючого автомату Мура на JK-тригерах для операції ділення першим способом.

# 3.1 Таблиця кодування сигналів

# 3.1.1 Функціональна схема:



*Рисунок 3.1.1* **-** *Функціональна схема.*

**3.1.2 Закодований мікроалгоритм**

Початок

Кінець

*SL3; SL2*

*i, W2*

*W2*

*x*

*STOP*

1

1

*R3; W2; W1*

*Рисунок 3.2. Закодованим мікроалгоритм пристрою ділення першим способом*

За закодованим мікроалгоритмом складемо таблицю:

*Таблиця 3.1 Таблиця кодування сигналів*

|  |  |
| --- | --- |
| Сигнали операційного автомата | Сигнали управляючого автомата |
| *R3, W1* | *Y1* |
| *SL3, SL2* | *Y2* |
| *W2* | *Y3* |
| *i* | *Y4* |
| *x* | *X1* |
| *STOP* | *X2* |

# 3.1.3 Закодований мікроалгоритм з управляючими сигналами автомата

Початок

Кінець

*Y2*

*Y3; Y4*

*Y3*

*X1*

*X2*

1

1

*Y1; Y3*

Z1

Z2

Z3

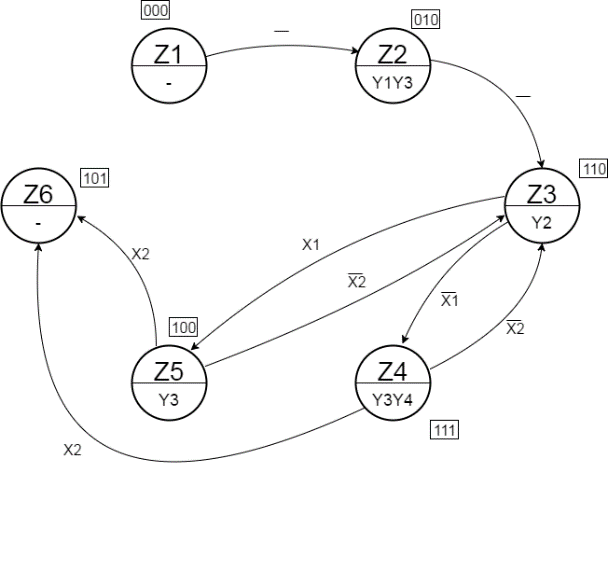
Z5

Z4

Z6

*Рисунок 3.3. Змістовний мікроалгоритм пристрою ділення першим способом з управляючими сигналами автомата*

**3.1.3 Граф управляючого автомата Мура з кодами вершин:**

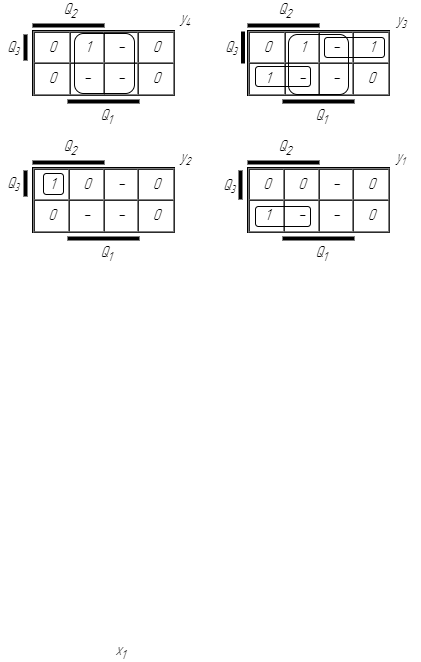
**

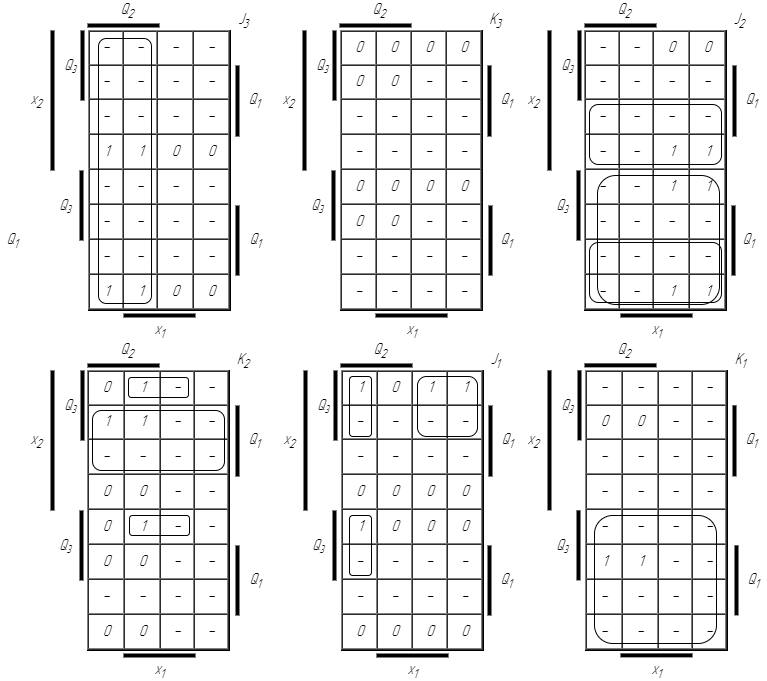
# 3.4 Таблиця переходів циклічного автомата на JK-тригерах

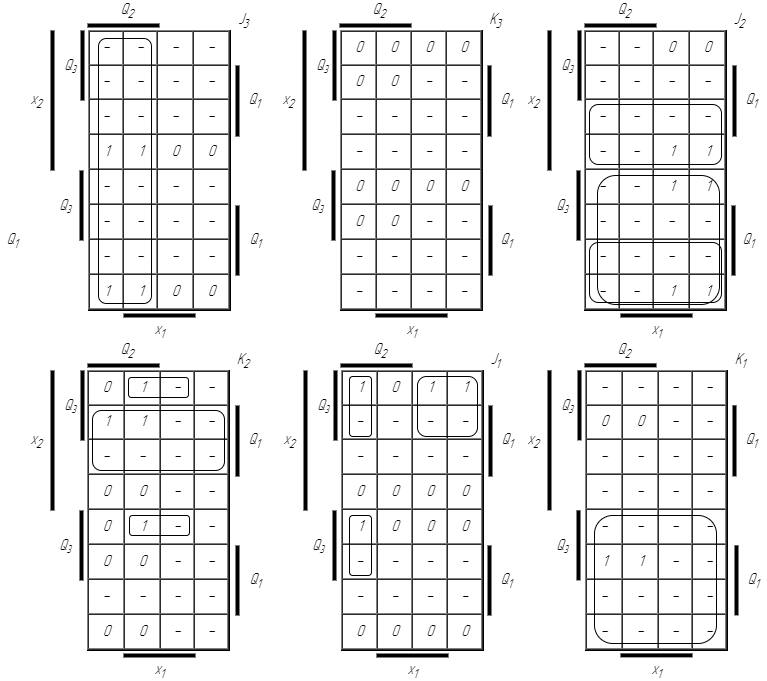
*Таблиця 3.2 – Таблиця переходів*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Перехід | Q3Q2Q1 | Q3Q2Q1 | x1x2 | y4y3y2y1 | J3K3 | J2K2 | J1K1 |
| *z1z2* | 000 | 010 | -- | 0000 | 0- | 1- | 0- |
| *z2 z3* | 010 | 110 | -- | 0101 | 1- | -0 | 0- |
| *z3 z4* | 110 | 111 | 0- | 0010 | -0 | -0 | 1- |
| *z3 z5* | 110 | 100 | 1- | 0010 | -0 | -1 | 0- |
| *z4 z3* | 111 | 110 | -0 | 1100 | -0 | -0 | -1 |
| *z4 z6* | 111 | 101 | -1 | 1100 | -0 | -1 | -0 |
| *z5 z3* | 100 | 110 | -0 | 0100 | -0 | 1- | 0- |
| *z5 z6* | 100 | 101 | -1 | 0100 | -0 | 0- | 1- |

# 3.5 Мінімізація функцій тригерів







Y4 = Q1

Y3 = Q1∨Q3Q2∨Q3Q2

Y2 = Q3Q2Q1

Y1 = Q3Q2

J3 = Q2

K3 = 0

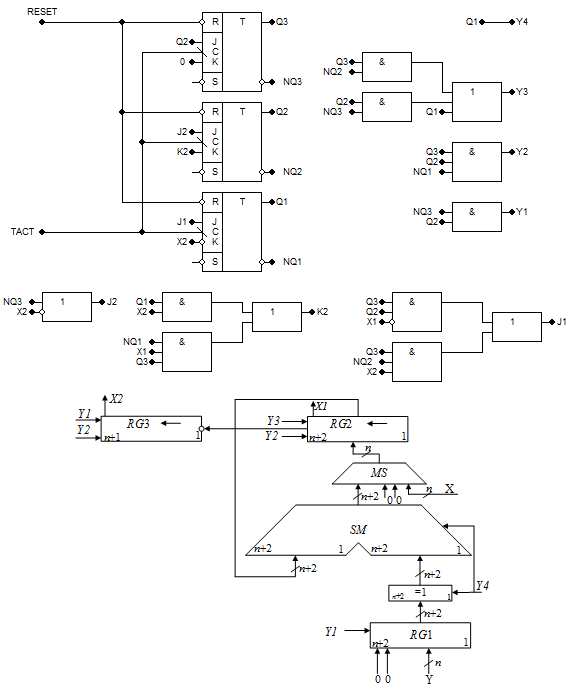
J2=Q3∨X2

K2 = Q1X2∨Q3Q1X1

J1 = Q3Q2X1∨Q3Q2X2

K1 = X2

# 3.6 Функціональна схема автомата



*Рисунок 3.5 - Функціональна схема*

#### Висновок

У даній розрахунковій роботі було виконано операції з числами в двійковому коді з плаваючою комою, а саме: множення чотирма способами, ділення двома способами, додавання та обчислення кореня. Для операції додавання було побудовано управляючий автомат Мура на JK-тригерах і елементах булевого базису. Зроблено мінімізацію функцій тригерів і в середовищі AFDK побудована функціональна схема автомата.

|  |  |
| --- | --- |
| ***Використана література:*** | |
| 1 | Жабін В.І., Жуков І.А., Клименко І.А.,Ткаченко В.В. Прикладна теорія цифрових автоматів: Навчальний посібник.*–*К.: Книжкове вид-во НАУ, 2009. *–* 360 с. |

Під час виконання даної розрахункової роботи я повторила для себе матеріал курсу «Комп’ютерна логіка - 1», а також закріпила знання з курсу «Комп’ютерна логіка - 2» (Комп’ютерна арифметика).