НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра обчислювальної техніки

РОЗРАХУНКОВА РОБОТА

по курсу „Комп'ютерна логіка-2”

Виконала: Гарасимович Галина Володимирівна
Група ІО-44, Факультет ІОТ,
Залікова книжка № 4406

Номер технічного завдання 1000100110110

_____

(підпис керівника)

Київ – 2015 р.

Завдання

1. Числа X_2 і Y_2 в прямому коді записати у формі з плаваючою комою (з порядком і мантисою, а також з характеристикою та мантисою), як вони зберігаються у пам'яті. На порядок (характеристику) відвести 8 розрядів, на мантису 16 розрядів (з урахуванням знакових розрядів).

2. Виконати 8 операцій з числами X_2 і Y_2 з плаваючою комою (чотири способи множення, два способи ділення, додавання та добування кореня з Y_2). Номери операцій (для п.3) відповідають порядку переліку (наприклад, 1 – множення першим способом; 6 – ділення другим способом; 8 – добування кореня). Для обробки мантис кожної операції, подати:

2.1 теоретичне обґрунтування способу;

2.1 операційну схему;

2.2 змістовний мікроалгоритм;

2.3 таблицю станів регістрів (лічильника), довжина яких забезпечує одержання 15 основних розрядів мантиси результату;

2.4 функціональну схему з відображенням управляючих сигналів;

2.5 закодований мікроалгоритм (мікрооперації замінюються управл. сигналами);

2.6 граф управляючого автомата Мура з кодами вершин;

2.7 обробку порядків (показати у довільній формі);

2.8 форму запису нормалізованого результату з плаваючою комою в пам'ять.

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

3. Для операції з двійковим номером $x_3x_2x_1+1$ побудувати управляючий автомат Мура на тригерах (тип тригера вибрати самостійно) і елементах булевого базису.

Визначення та обгрунтування варіанту:

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

$$X_2 = -1x_{10}x_9 1x_8x_7x_6 1, x_5x_4 0x_3 1x_2x_1 \text{ і } Y_2 = +1x_{10}1x_9x_8, x_7x_6 1x_5x_4 0x_3x_2x_1 1,$$

де x_i - двійкові цифри номера залікової книжки у двійковій системі числення (x_1 - молодший розряд).

$4406_{10}=1000100110110_2$;

$X_2 = -1x_{10}x_9 1x_8x_7x_6 1, x_5x_4 0x_3 1x_2x_1$ = -10110011,1001110;

$Y_2 = +1x_{10}1x_9x_8, x_7x_6 1x_5x_4 0x_3x_2x_1 1$ = +10110,0111001101;

Основна частина:

Завдання №1

$X_{пк}$=1.10110011,1001110;

$Y_{пк}$= 0.10110,0111001101;

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

X_2:

0	0	0	0	1	0	0	0

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

Y_2:

0	0	0	0	0	1	0	1

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

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

$E = P + 2^m$,

m = 7;

$2^7 = 10000000_2$

E_x = 10000000 + 111 = 10000111

X_2:

1	0	0	0	0	1	1	1

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

E_y = 10000000 + 100 = 10000100

Y_2:

1	0	0	0	0	1	0	0

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

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:

$Z=YX=Yx_n2^{-n}+ Yx_{n-1}2^{-n+1}\ldots+ Yx_1 2^{-1} =$

$= ((..((0+Yx_n)\,2^{-1}+ Yx_{n-1})\,2^{-1}+\ldots+ Yx_i)\,2^{-1} +\ldots+ Yx_1)\,2^{-1};$

$Z=\sum_{i=1}^{n}(Z_{i-1} + Yx_{n-i+1})2^{-1};$

2.1.2 Операційна схема:

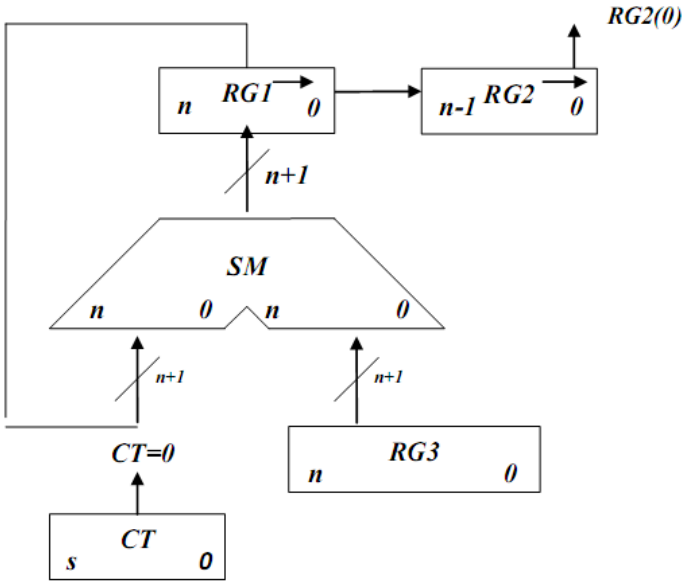


Рисунок 2.1.1- Операційна схема.

2.1.3 Змістовний мікроалгоритм:

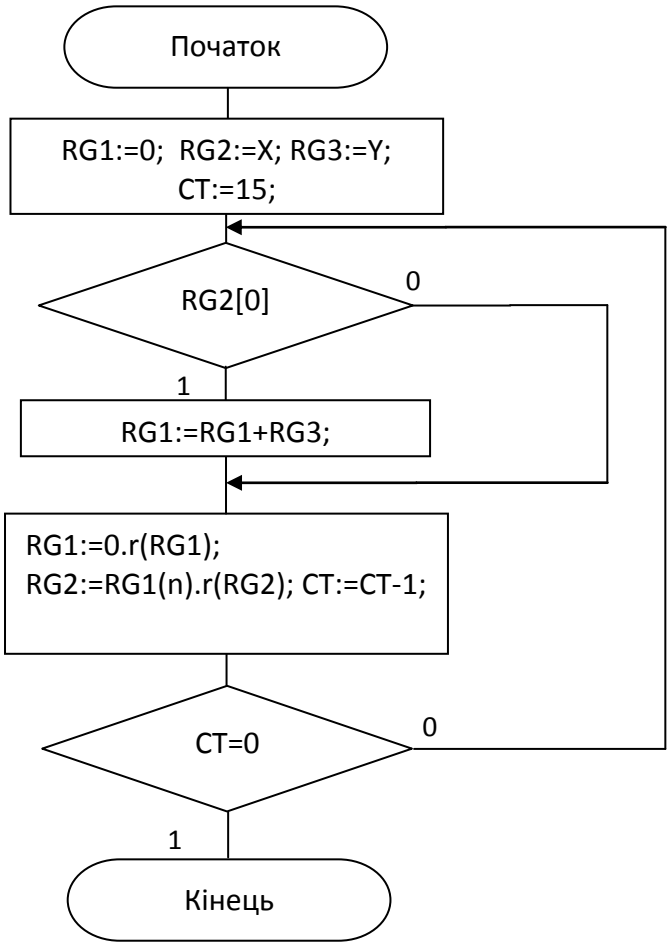


Рисунок 2.1.2 - Змістовний мікроалгоритм виконання операції множення першим способом.

2.1.4 Таблиця станів регістрів:

Таблиця 2.1.1-Таблиця станів регістрів для першого способу множення.

№	RG1	RG2	RG3	CT
пс	0	101100111001110	101100111001101	1111
1	0	010110011100111		1110
2	00010110011100110	101011001110011		1101
3	+ 00101100111001101 1000011010110011 0100001101011001	110101100111001		1100
4	+ 00101100111001101 1001110100100110 0100111010010011	011010110011100		1011
5	0010011101001001	101101011001110		1010
6	0001001110100100	110110101100111		1001
7	+ 00101100111001101 0110110101110001 0011011010111000	111011010110011		1000
8	+ 00101100111001101 1001000010000101 0100100001000010	111101101011001		0111
9	+ 00101100111001101 1010001000001111 0101000100000111	111110110101100		0110
10	0010100010000011	111111011010110		0101
11	0001010001000001	111111101101011		0100
12	+ 00101100111001101 0110111000001110 0011011100000111	011111110110101		0011
13	+ 00101100111001101 1001000011010100 0100100001101010	001111111011010		0010
14	0010010000110101	000111111101101		0001
15	+ 00101100111001101 0111111000000010 0011111100000001	000011111110110		0000

2.1.5 Функціональна схема:

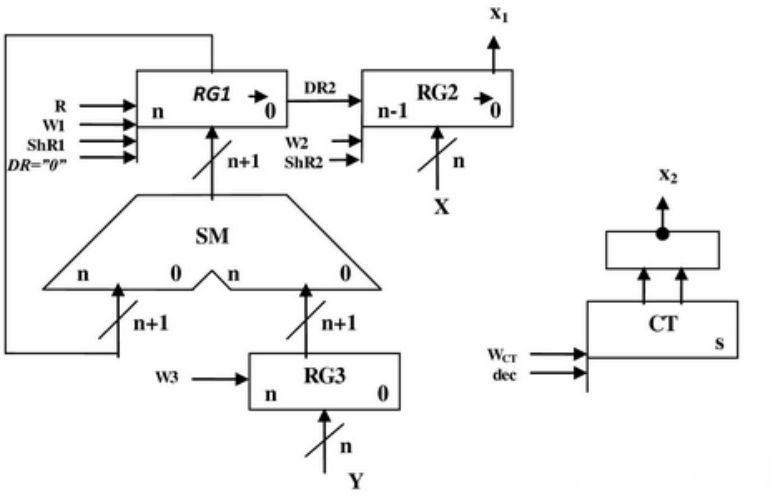


Рисунок 2.1.3- Функціональна схема.

2.1.6 Закодований мікроалгоритм

Таблиця 2.1.2-Таблиця кодування операцій і логічних умов.

Кодування мікрооперацій		Кодування логічних умов	
МО	УС	ЛУ	Позначення
G1:=0	R	RG2[0]	X1
RG2:=X	W2	CT=0	X2
RG3:=Y	W3		
CT:=15	W_{CT}		
RG1:=RG1+RG3	W1		
RG1:=0.r(RG1)	ShR1		
RG2:=RG1[0].r(RG2)	ShR2		
CT:=CT-1	dec		

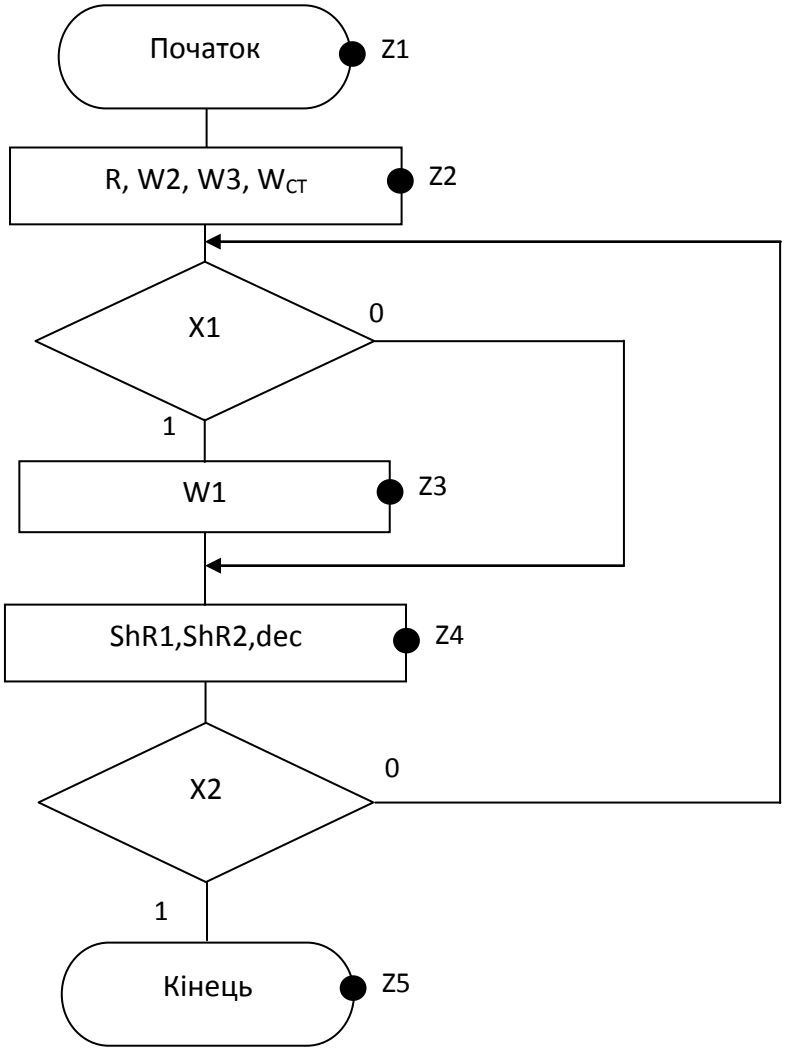


Рисунок 2.1.4-Закодований мікроалгоритм.

2.1.7 Граф управляючого автомата Мура з кодами вершин:

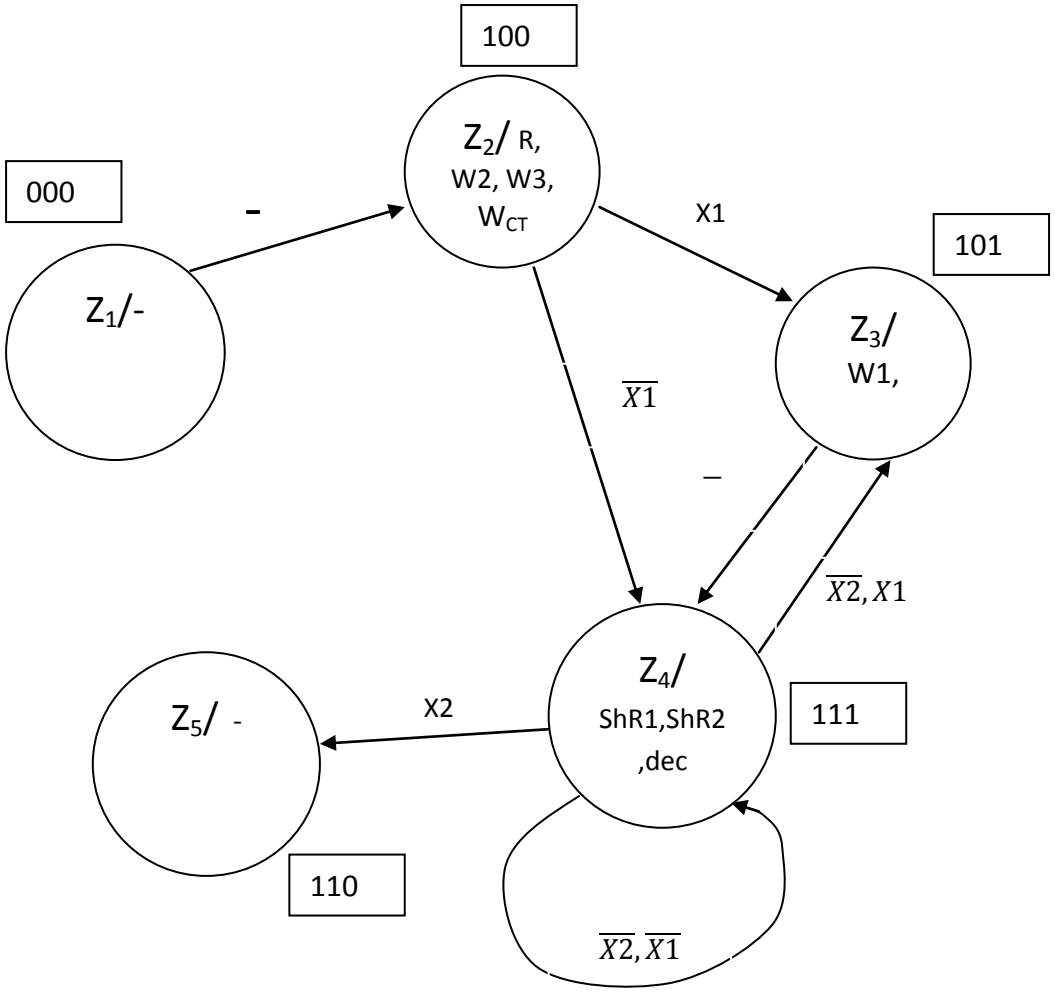


Рисунок 2.1.5-Граф автомата Мура

2.1.8 Обробка порядків:

Порядок добутку буде дорівнювати сумі порядків множників з урахуванням знаку порядків: $P_z = P_x + P_y$;

P_x=8; P_y=5; P_z=13_{10}=1101_2

2.1.9 Нормалізація результату:

Отримали результат: 0111111000000010

Знак мантиси: 1 ⊕ 0 = 1.

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

Порядок зменшуємо на 1:

1111110000000100; P_z=12;

Запишемо нормалізований результат:

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

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.

$Z=YX_n2^{-n}+ YX_{n-1}2^{-n+1}\ldots+ YX_1 2^{-1};$

$Z=((0+ YX_n2^{-n})+ YX_{n-1}2^{-n+1})\ldots+ YX_1 2^{-1};$

$Z=\sum_{i=1}^{n} Z_{i-1} + YX_{n-i+1}2^{-n+i-1};$

$Z_0 = 0;$

$Y_0 = 0$

2.2.2 Операційна схема:

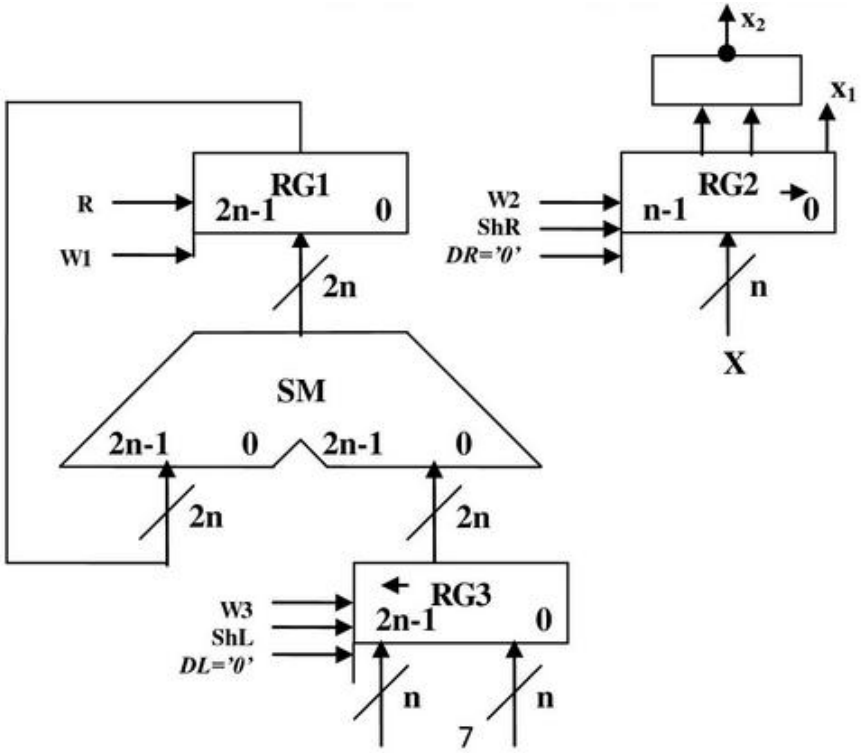


Рисунок 2.2.1- Операційна схема

2.2.3 Змістовний мікроалгоритм:

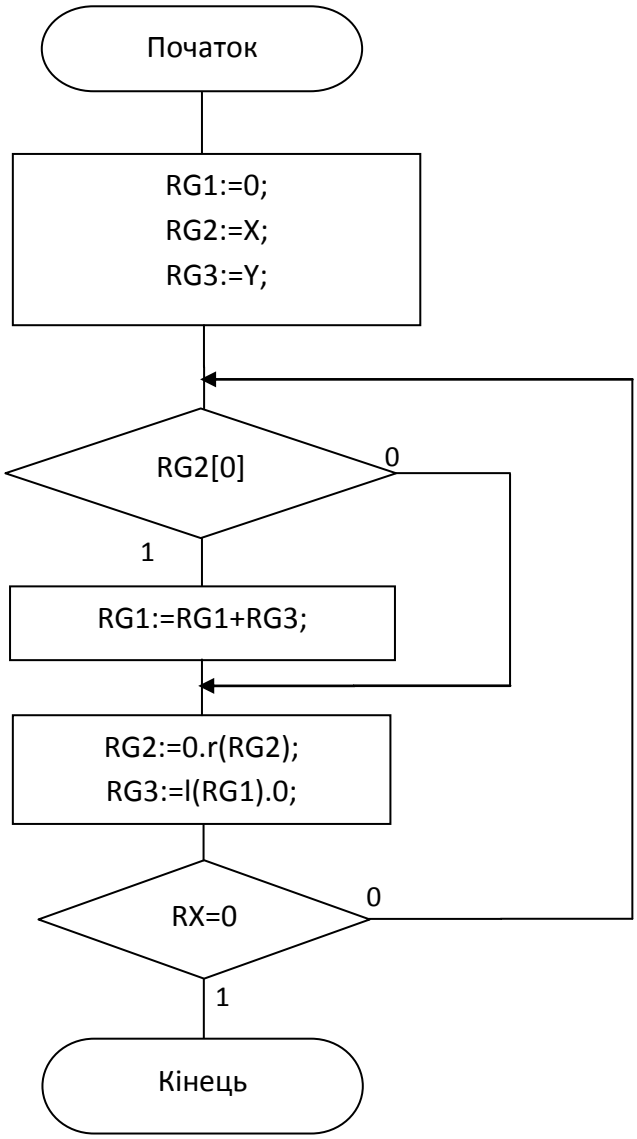


Рисунок 2.2.2 - Змістовний мікроалгоритм.

2.2.4 Таблиця станів регістрів:

Таблиця 2.2.1-Таблиця станів регістрів.

№	RG1	RG3←	RG2→
пс	0	00000000000000001011001110011101	101100111001110
1	0	00000000000000001011001110011010	010110011100111
2	00000000000000001011001110011010	00000000000000001011001110011010 0	001011001110011
3	+ 00000000000000001011001110011010 0 00000000000010000110101100111 0	00000000000000101100111001101000	000101100111001
4	+ 00000000000000101100111001101000 0 00000000000100111010010011011 0	00000000000001011001110011010000	000010110011100
5	00000000000100111010010011011 0	00000000000101100111001101000000	000001011001110

6	00000000000100111010010011 0110	00000000001011001110011010000000	000000101100111
7	+ 00000000001011001110011010000000 00000000011011010111000111 0110	00000000101100111001101000000 0	000000010110011
8	+ 00000000101100111001101000000000 00000001001000010000101111 0110	00000000101100111001101000000000	000000001011001
9	+ 00000000101100111001101000000000 00000010100010000011111111 0110	00000001011001110011010000000000	000000000101100
10	00000010100010000011111111 0110	00000010110011100110100000000000	000000000010110
11	00000010100010000011111111 0110	00000101100111001101000000000000	000000000001011
12	+ 00000101100111001101000000000000 00001101110000011101111111 0110	00001011001110011010000000000000	000000000000101
13	+ 00001011001110011010000000000000 00100100001101010001111111 0110	00010110011100110100000000000000	000000000000010
14	00100100001101010001111111 0110	00101100111001101000000000000000	000000000000001
15	+ 00101100111001101000000000000000 01111110000000100001111111 0110	01011001110011010000000000000000	000000000000000

2.2.5 Функціональна схема:

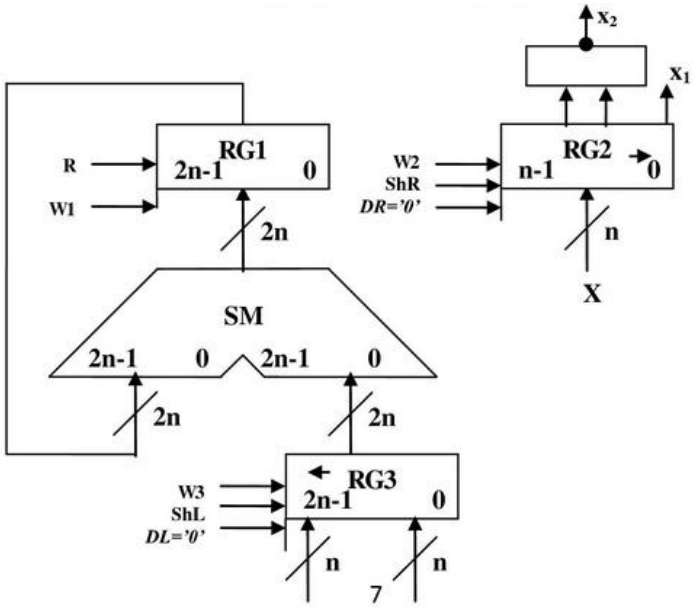


Рисунок 2.2.3- Функціональна схема.

2.2.6 Закодований мікроалгоритм

Таблиця 2.2.2-Таблиця кодування операцій і логічних умов.

Кодування мікрооперацій		Кодування логічних умов	
МО	УС	ЛУ	Позначення
RG1:=0 RG2:=X	R W2	RG2[0] RG2=0	X1 X2

RG3:=Y RG1:=RG1+RG3 RG2:=0.r(PG2) RG3:=l(RG3).0	W3 W1 ShR ShL		

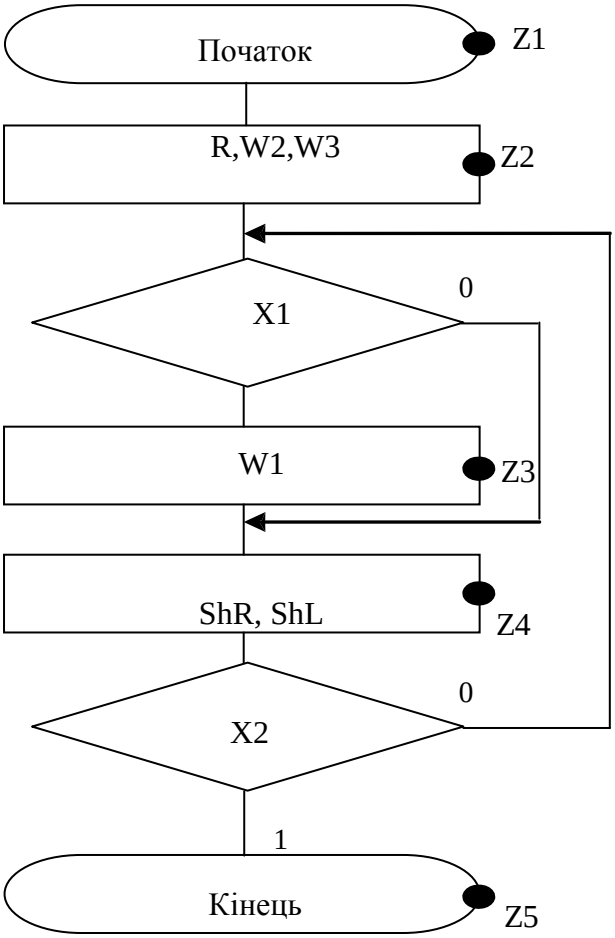


Рисунок 2.2.4-Закодований мікроалгоритм.

2.2.7 Граф управляючого автомата Мура з кодами вершин:

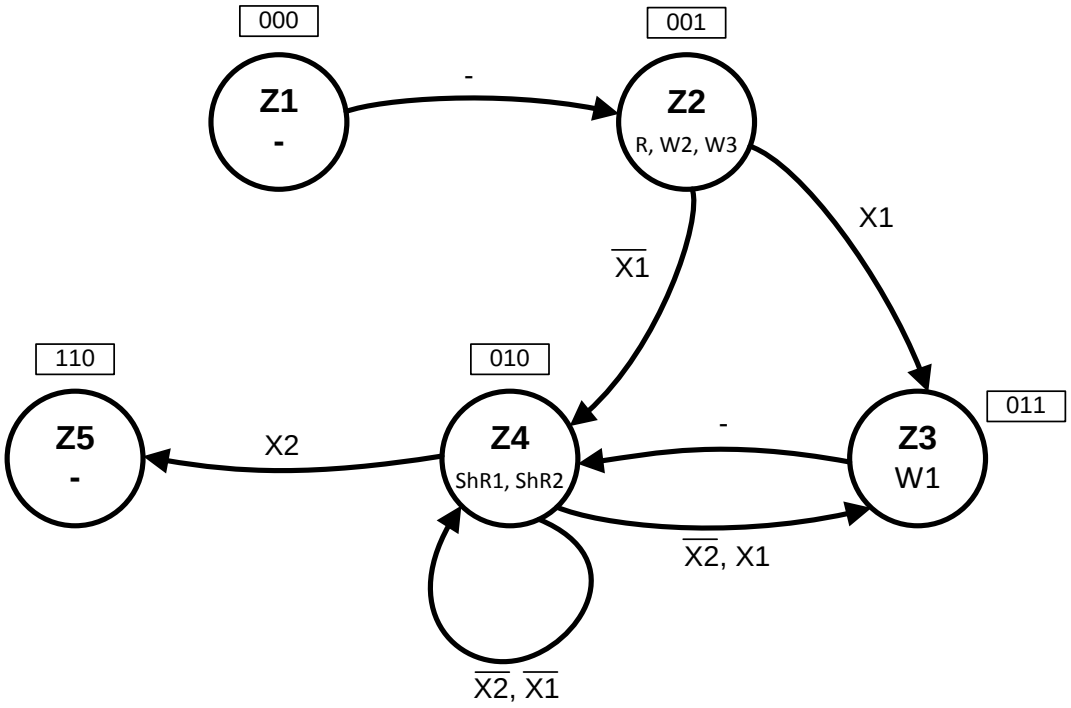


Рисунок 2.2.5 - Граф автомата Мура

2.2.8 Обробка порядків:

Порядок добутку буде дорівнювати сумі порядків множників з урахуванням знаку порядків: $P_z = P_x + P_y$;

P_x=8; P_y=5; P_z=13_{10}=1101_2

2.2.9 Нормалізація результату:

Отримали результат: 0111111000000010

Знак мантиси: 1 ⊕ 0 = 1.

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

Порядок зменшуємо на 1:

1111110000000100; P_z=12;

Запишемо нормалізований результат:

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

2.3 Третій спосіб множення.

2.3.1Теоретичне 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.

$Z=Y X_n 2^{-n} + Y X_{n-1} 2^{-n+1} \ldots + Y X_1 2^{-1}$;

$Z= Y X_n 2^{-n} + 2(Y X_{n-1} 2^{-n} + 2(Y X_{n-2} 2^{-n} \ldots + 2 Y X_1 2^{-n}))$;

$Z=\sum_{i=1}^{n} 2Z_{i-1} + Y X_i 2^{-n}$;

$Z_0 = 0$;

$Y_0 = 0$

2.3.2 Операційна схема:

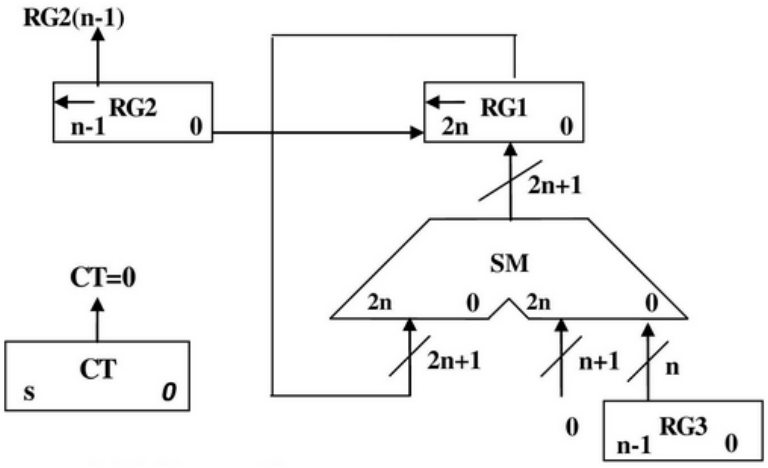


Рисунок 2.3.1 - Операційна схема

2.3.3 Змістовний мікроалгоритм:

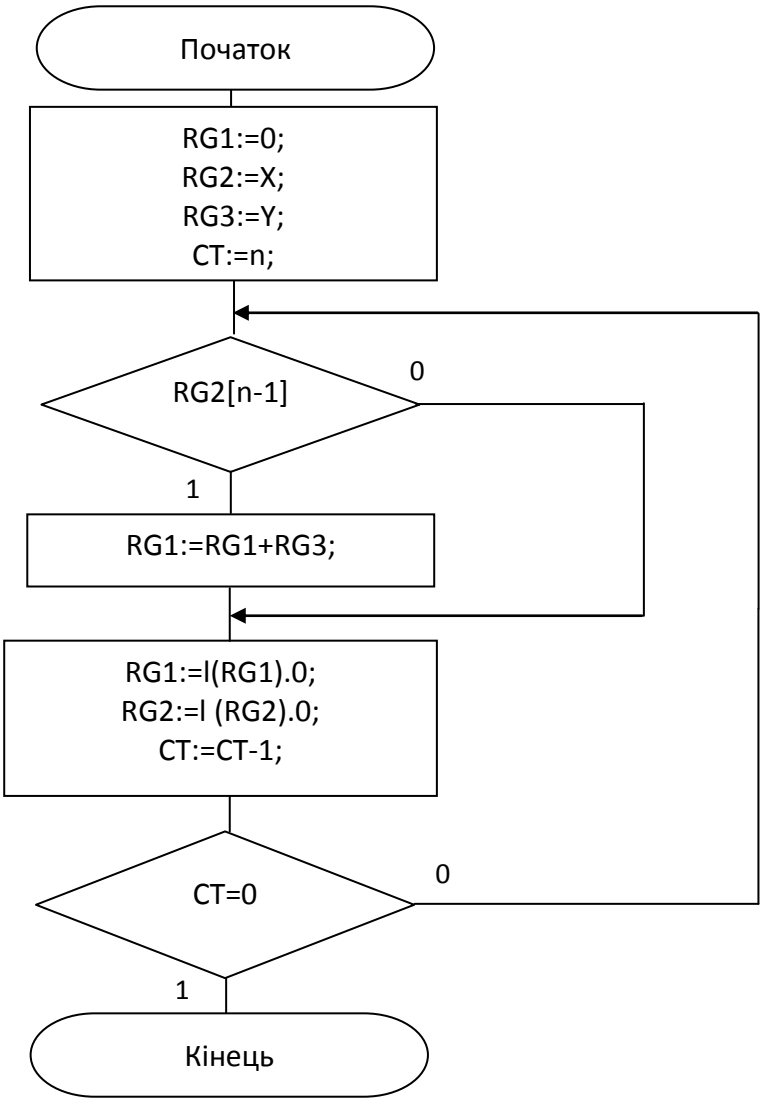


Рисунок 2.3.2 - Змістовний мікроалгоритм.

2.3.4 Таблиця станів регістрів:

Таблиця 2.3.1- Таблиця станів регістрів

№	RG1←	RG2←	RG3	CT
пс	00000000000000000000000000000000	101100111001110	101100111001101	1111
1	00000000000000001011001110011010	011001110011100		1110
2	00000000000000010110011100110100	110011100111000		1101
3	+ 00000000000000000101100111001101 00000000000000111000001000000001 00000000000001110000010000000010	100111001110000		1100
4	+ 00000000000000000101100111001101 00000000000001111011011110011111 00000000000011110110111100111110	001110011100000		1011
5	00000000001111011011110011110	011100111000000		1010
6	00000000011110110111001111000	111001110000000		1001
7	+ 00000000000000000101100111001101	110011100000000		1000

	00000000011111001110000100010100000000111110011100001000101			
8	+ 00000000000000001011001110011010000000111110110010100101011100000000111110110010100101011010	100111000000000		0111
9	+ 00000000000000001011001110011010000000111110111010111001111011000000011111011101111001111011010	001110000000000		0110
10	00000111110111101110011110110000	011100000000000		0101
11	00001111101111011100111101100000	111000000000000		0100
12	+ 00000000000000001011001110011010000111110111111001101101001010000111110111111100110110100101	110000000000000		0011
13	+ 00000000000000001011001110011010000111110111111111010100010111000111110111111111101010001011100	100000000000000		0010
14	+ 00000000000000001011001110011010011111100000001000001111111011000111110000000100001111111010110	000000000000000		0001
15	**11111000000010000111111101100**	000000000000000		0000

2.3.5 Функціональна схема:

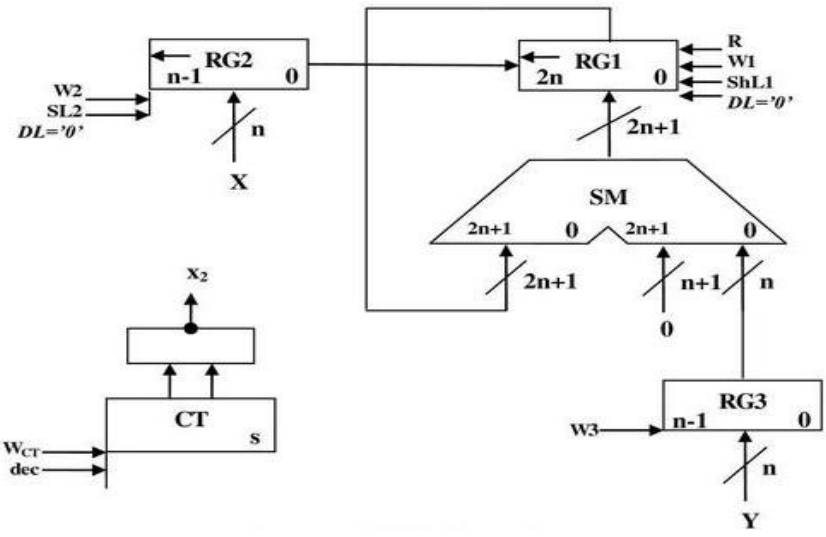


Рисунок 2.3.3 - Функціональна схема.

2.3.6 Закодований мікроалгоритм:

Таблиця 2.3.2-Таблиця кодування операцій і логічних умов.

Кодування мікрооперацій		Кодування логічних умов	
МО	УС	ЛУ	Позначення
RG1:=0	R	RG2[n-1]	X1
RG2:=X	W2	CT=0	X2
RG3:=Y	W3		
CT:=15	W_{CT}		
RG1:=RG1+RG3	W1		
RG1:=l(RG1).0	ShL1		
RG2:=l(RG2).0	ShL2		
CT:=CT-1	dec		

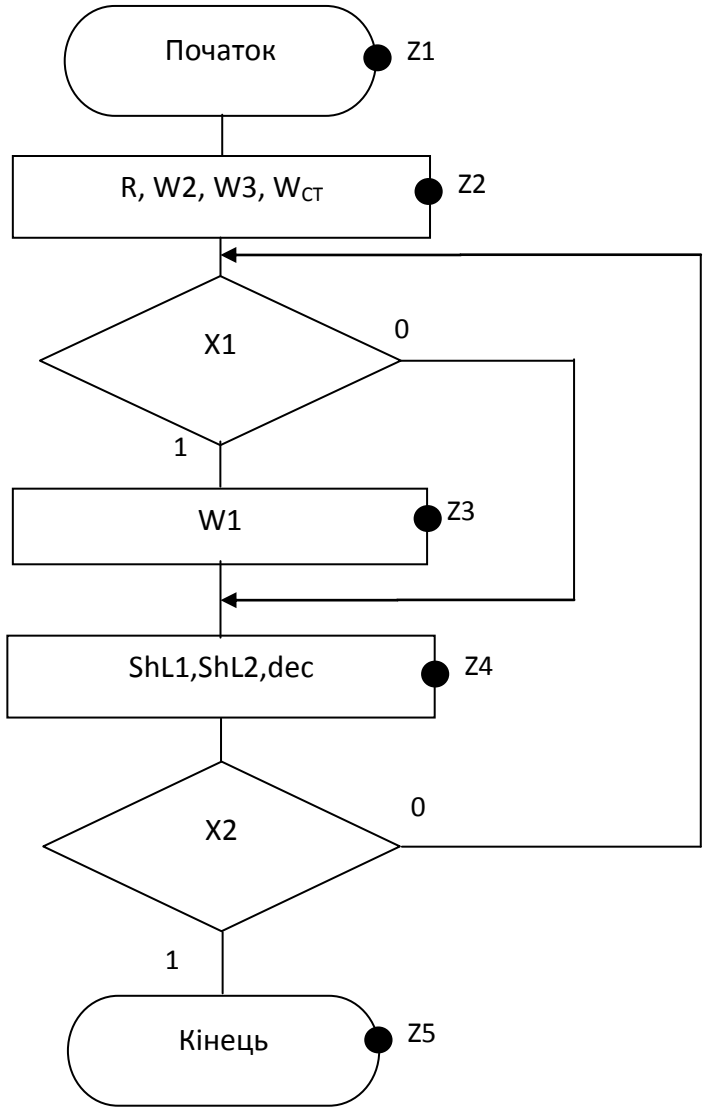


Рисунок 2.3.4-Закодований мікроалгоритм.

2.3.7 Граф управляючого автомата Мура з кодами вершин:

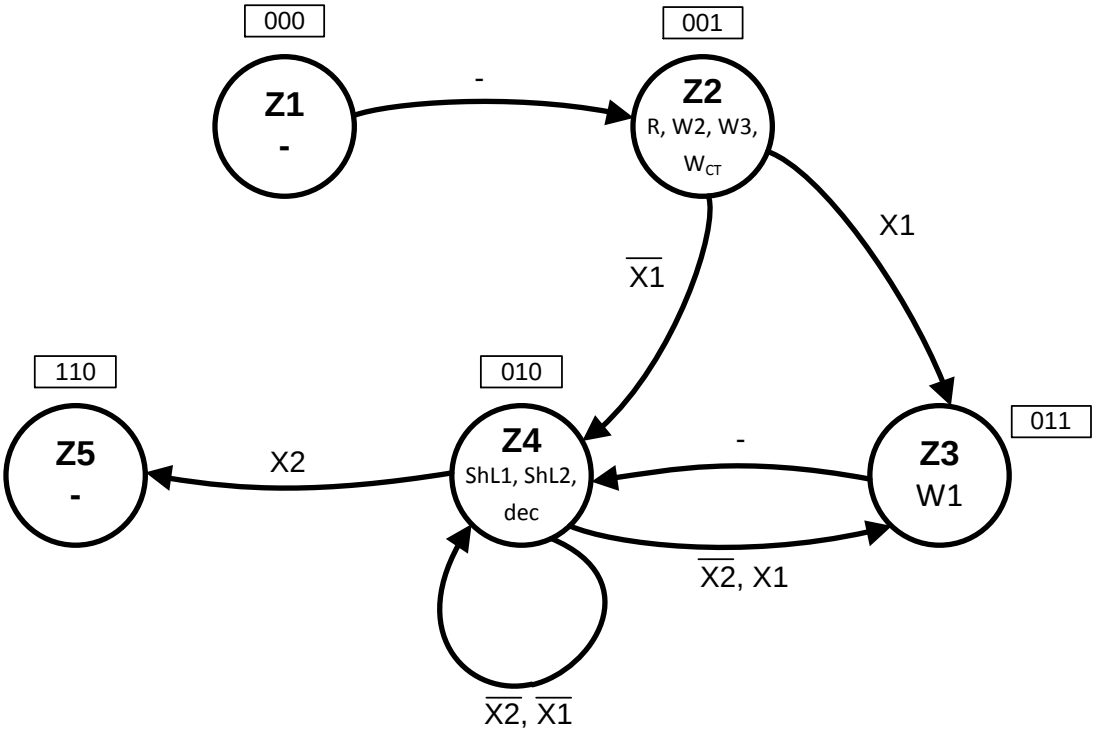


Рисунок 2.3.5 - Граф автомата Мура

2.3.8 Обробка порядків:

Порядок добутку буде дорівнювати сумі порядків множників з урахуванням знаку порядків: $P_z = P_x + P_y$;

P_x=8; P_y=5; P_z=13_{10}=1101_2

2.3.9 Нормалізація результату:

Отримали результат: 0111111000000010

Знак мантиси: $1 \oplus 0 = 1$.

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

Порядок зменшуємо на 1:

1111110000000100; P_z=12;

Запишемо нормалізований результат:

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

2.4 Четвертий спосіб множення.

2.4.1Теоритичне 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.

$Z = Y \cdot x_n \cdot 2^{-n} + Y \cdot x_{n-1} \cdot 2^{-n+1} + \cdots + Y \cdot x_1 \cdot 2^{-1}$.

$Z = ((\ldots((0 + Y \cdot 2^{-1}x_1) + Y \cdot 2^{-2}x_2) + \cdots + Y \cdot 2^{-k}x_k) + \cdots + Y \cdot 2^{-k}x_k)$.

$Z_i = Z_{i-1} + 2^{-1}Y_{i-1} \cdot x_i$ з початковими значеннями i=1, $Y_0=2^{-1}Y$, $Z_0=0$.

2.4.2 Операційна схема:

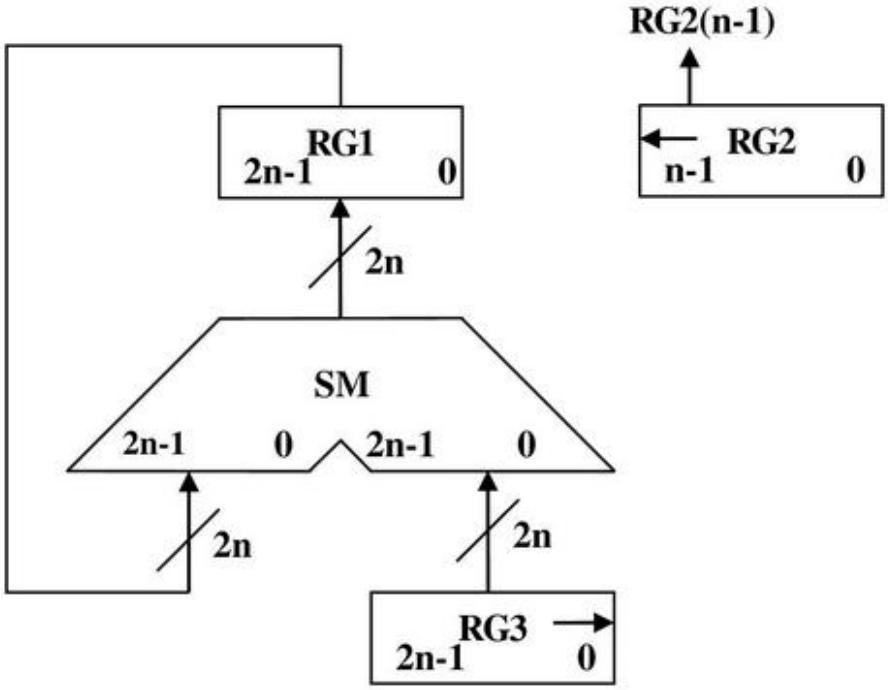


Рисунок 2.4.1- Операційна схема

2.4.3 Змістовний мікроалгоритм:

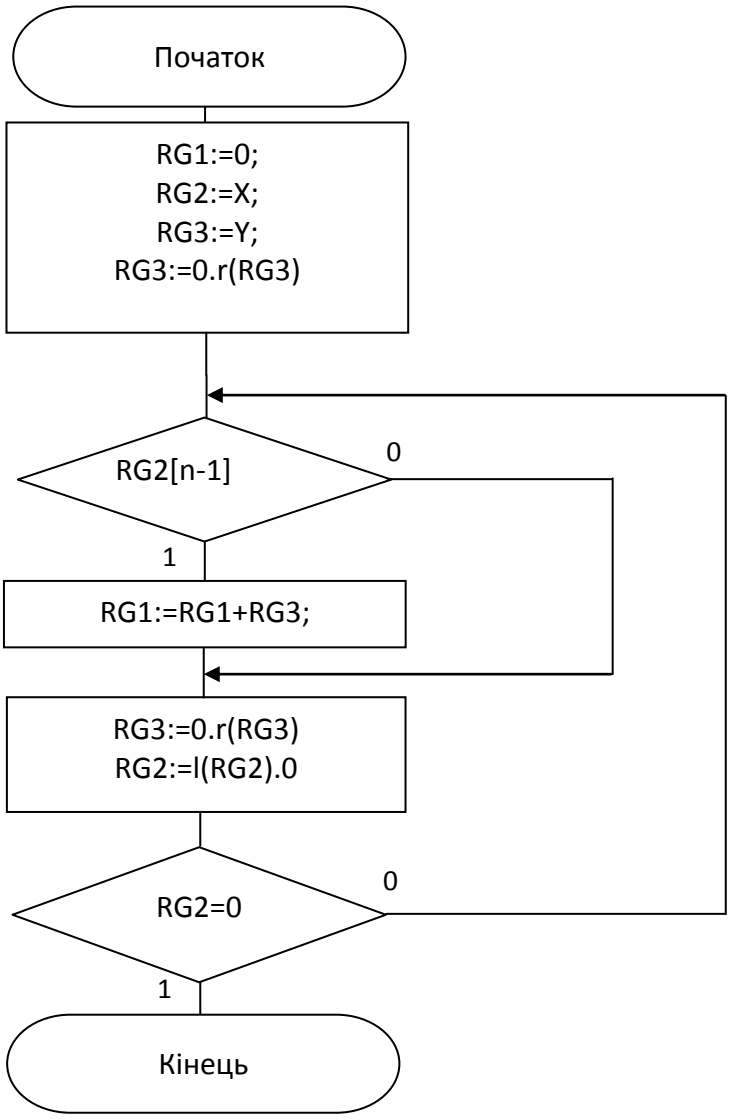


Рисунок 2.4.2 - Змістовний мікроалгоритм.

2.4.4 Таблиця станів регістрів:

Таблиця 2.4.1- Таблиця станів регістрів

№	RG1	RG3 →	RG2 ←
ПС	00000000000000000000000000000000	00101100111001101000000000000000	101100111001110
1	00101100111001101000000000000000	00010110011100110100000000000000	011001110011100
2	00101100111001101000000000000000	00001011001110011010000000000000	110011100111000
3	+ 00001011001110011010000000000000 01110000010000001000000000000	00000101100111001101000000000000	100111001110000
4	+ 00000101100111001101000000000000 01111011011110011110000000000	00000010110011100110100000000000	001110011100000
5	01111011011110011110000000000000	00000001011001110011010000000000	011100111000000
6	01111011011110011110000000000000	00000000101100111001101000000000	111001110000000

7	+ 000000001011001110011010000000 01111100111000010001010000000	000000000101100111001101000000	110011100000000
8	+ 000000000101100111001101000000 01111101100101001010111000000	000000000010110011100110100000	100111000000000
9	+ 000000000010110011100110100000 01111101111011100111011000000	000000000001011001110011010000	001110000000000
10	01111101111011100111011000000	000000000000101100111001101000	011100000000000
11	01111101111011100111011000000	000000000000010110011100110100	111000000000000
12	+ 000000000000010110011100110100 01111101111110011011010010100 0	000000000000001011001110011010	110000000000000
13	+ 000000000000001011001110011010 01111101111111110101000101110 0	000000000000000101100111001101	100000000000000
14	+ 000000000000000101100111001101 01111110000000100001111111011 0	000000000000000010110011100110	000000000000000
15	**011111100000001000011111110110**	000000000000000001011001110011 0	000000000000000

2.4.5Функціональна схема:

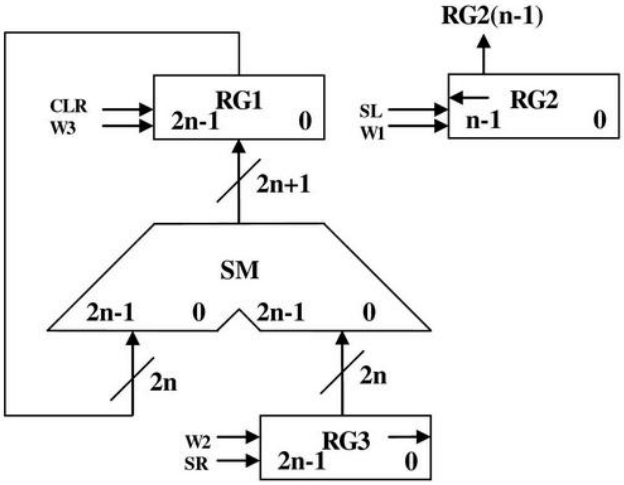


Рисунок 2.4.3 - Функціональна схема.

2.4.6 Закодований мікроалгоритм

Таблиця 2.4.2-Таблиця кодування операцій і логічних умов.

Кодування мікрооперацій		Кодування логічних умов	
МО	УС	ЛУ	Позначення
RG1:=0	R	RG2[n-1]	X1
RG2:=X	W2	RG2=0	X2
RG3:=Y	W3		
RG1:=RG1+RG3	W1		
RG3:=0.r(RG3)	ShR		
RG2:=l(RG2).0	ShL		

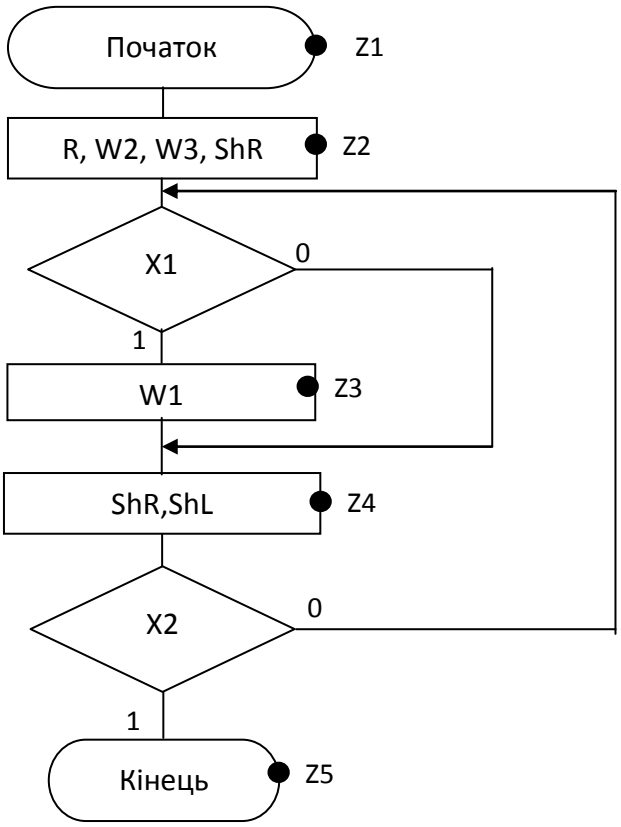


Рисунок 2.4.4-Закодований мікроалгоритм.

2.4.7 Граф управляючого автомата Мура з кодами вершин:

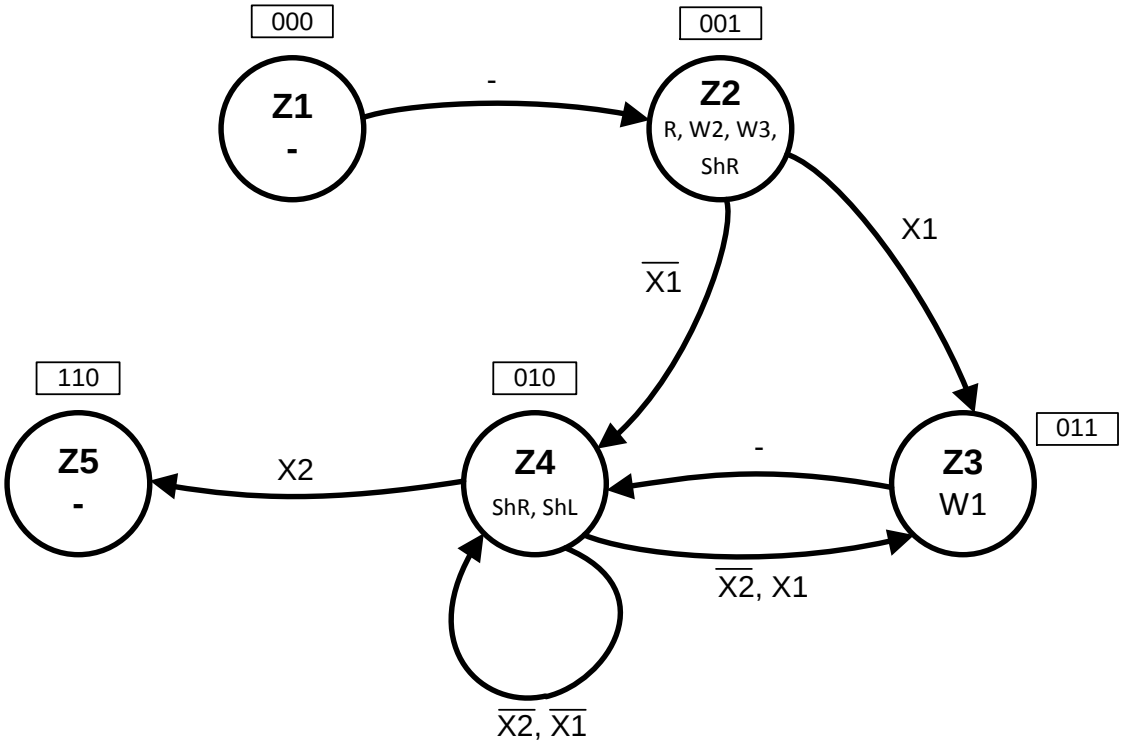


Рисунок 2.4.5 - Граф автомата Мура

2.4.8 Обробка порядків:

Порядок добутку буде дорівнювати сумі порядків множників з урахуванням знаку порядків: $P_z = P_x + P_y$;

P_x=8; P_y=5; P_z=13_{10}=1101_2

2.4.9 Нормалізація результату:

Отримали результат: 0111111000000010

Знак мантиси: $1 \oplus 0 = 1$.

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

Порядок зменшуємо на 1:

1111110000000100; P_z=12;

Запишемо нормалізований результат:

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

2.5. Першиий спосіб ділення.

2.5.1Теоритичне обґрунтування першого способу ділення:

Нехай ділене X 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 X). 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:

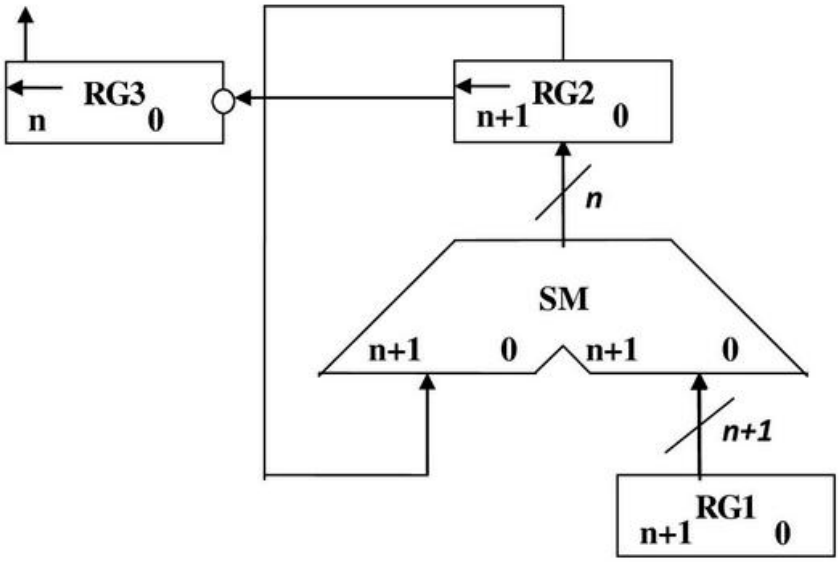


Рисунок 2.5.1- Операційна схема

2.5.3 Змістовний мікроалгоритм:

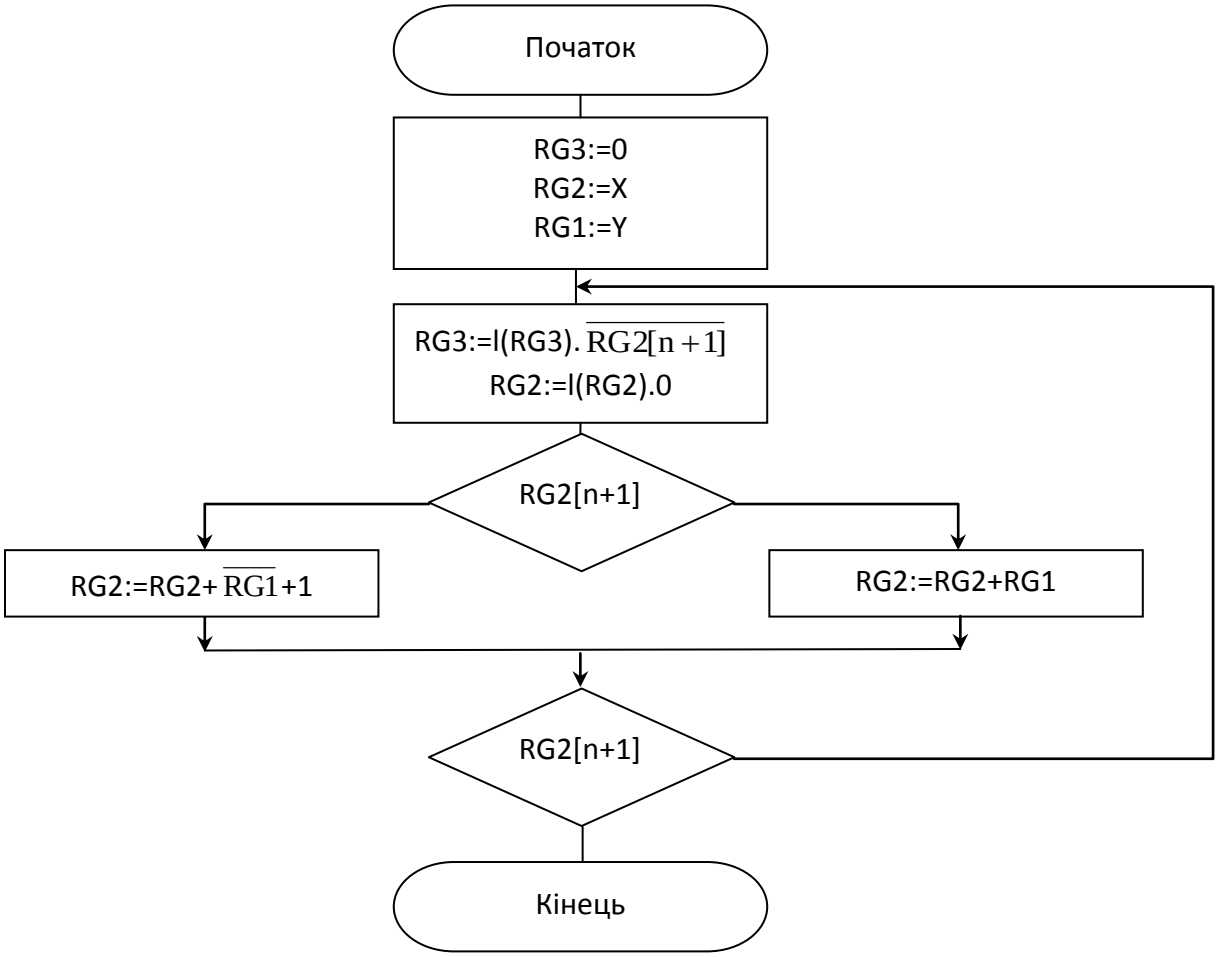


Рисунок 2.5.2-Змістовний мікроалгоритм

2.5.4 Таблиця станів регістрів:

№	RG3(Z)	RG2(X)	RG1(Y)
пс	0000000000000000	00101100111001110	101100111001101
1	0000000000000001	01011001110011100 + 11101001100011010 01000011010110110	
2	0000000000000011	10000110101101100 + 00010110011100 1101 10110011100111001	
3	0000000000000110	01100111001110010 + 11101001100011010 01010000110001100	
4	0000000000001101	10100001100011000 + 00010110011100 1101 11001110011100101	
5	0000000000011010	10011100111001010 + 00010110011100 1101 11001001110010111	
6	0000000000110100	10010011100101110 + 00010110011100 1101 11000000011111011	
7	0000000001101000	10000000111110110 + 00010110011100 1101 10101101111000011	
8	0000000011010000	01011011110000110 + 11101001100011010 01000101010100000	
9	0000000110100001	10001010101000000 + 00010110011100 1101 10110111100001101	
10	0000001101000010	01101111000011010 + 11101001100011010 01011000100110100	
11	0000011010000101	10110001001101000 +	

		000101100111001101 11011110000110101	
12	0000110100001010	10111100001101010 + 000101100111001101 11101001000110111	
13	0001101000010100	11010010001101110 + 000101100111001101 11111111000111011	
14	0011010000101000	11111110001110110 + 000101100111001101 00101011001000011	
15	0110100001010001	01010110010000110 + 11101001100011010 00111111110100000	
16	**1101000010100011**	01111111101000000 + 11101001100011010 01101001001011010	

2.5.5 Функціональна схема:

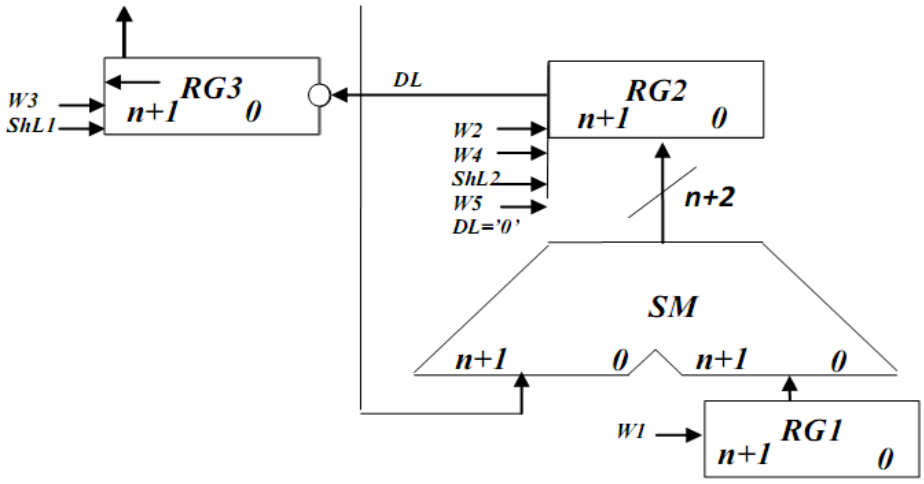


Рисунок 2.5.3 – Функціональна схема

2.5.6 Закодований мікроалгоритм

Таблиця 2.5.2-Таблиця кодування операцій і логічних умов.

Кодування мікрооперацій		Кодування логічних умов	
МО	УС	ЛУ	Позначення
RG3:=0 RG2:=X; RG1:=Y;	W3 W2 W1	RG2[n-1] RG2=0	X1 X2

RG3:=l(RG3).$\overline{RG2[n+1]}$ RG2:=l(RG2).0 RG2:=RG2+$\overline{RG1}$+1 RG2:=RG2+RG1	ShL1 ShL2 W4 W5		

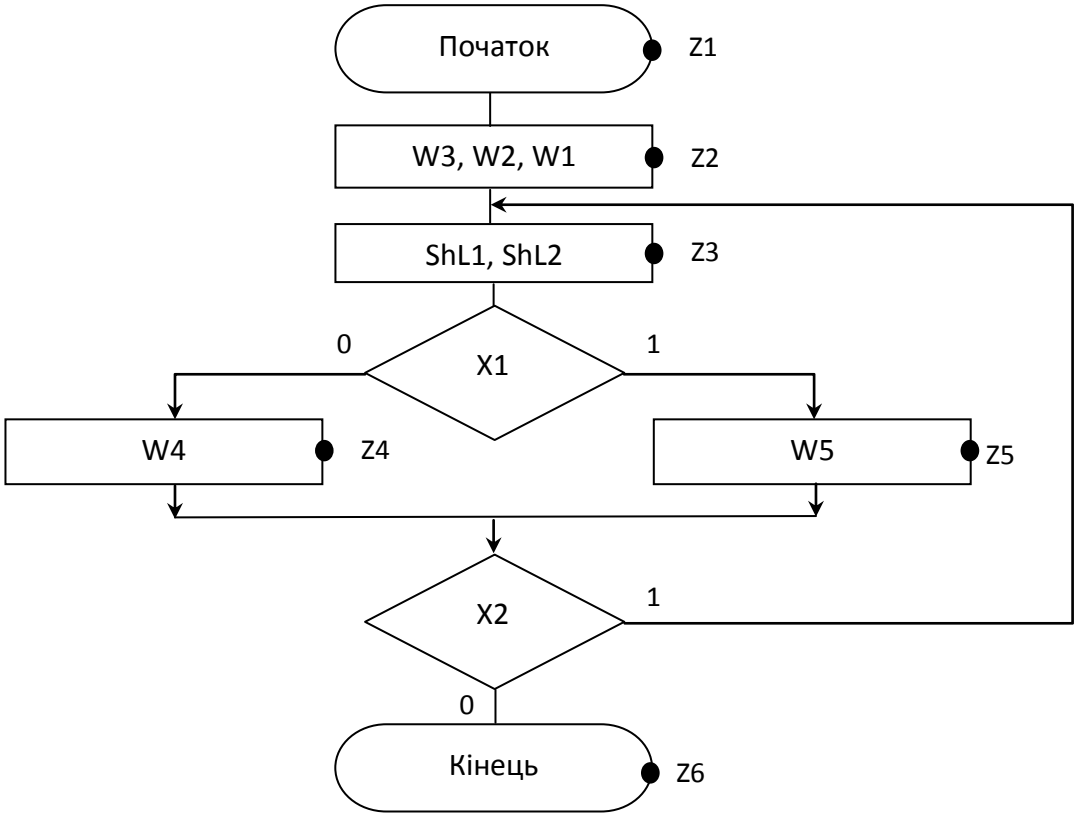


Рисунок 2.5.4-Закодований мікроалгоритм.

2.5.7 Граф управляючого автомата Мура з кодами вершин:

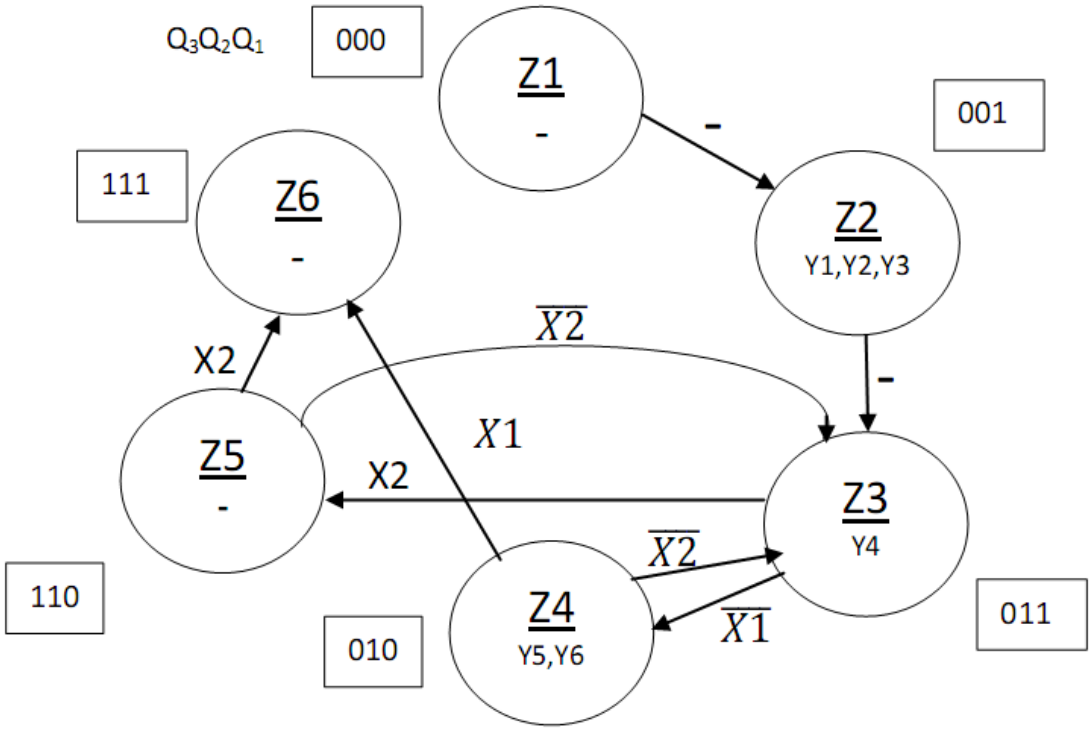


Рисунок 2.5.5 - Граф управляючого автомата.

2.5.8 Обробка порядків:

Порядок частки буде дорівнювати: $P_z = P_x - P_y$;

В моєму випадку P_x=8; P_y=5; P_z=3;

2.5.8 Нормалізація результату:

Отримали результат: **1101000010100011**

Знак мантиси: 1 ⊕ 0 = 1.

Нормалізація мантиси не потрібна.

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

2.6. Другий спосіб ділення.

2.6.1Теоритичне обгрунтування другого способу ділення:

Нехай ділене X 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

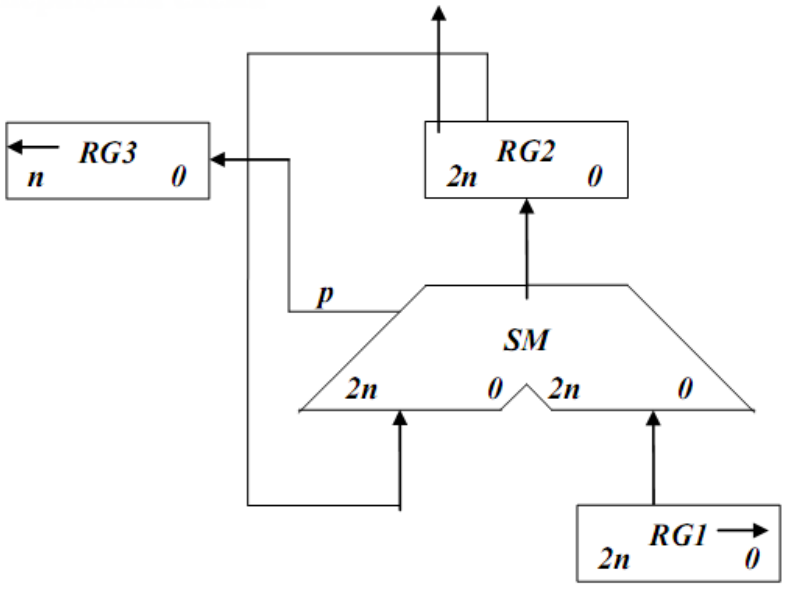


Рисунок 2.6.1-Операційна схема

2.6.3 Змістовний мікроалгоритм

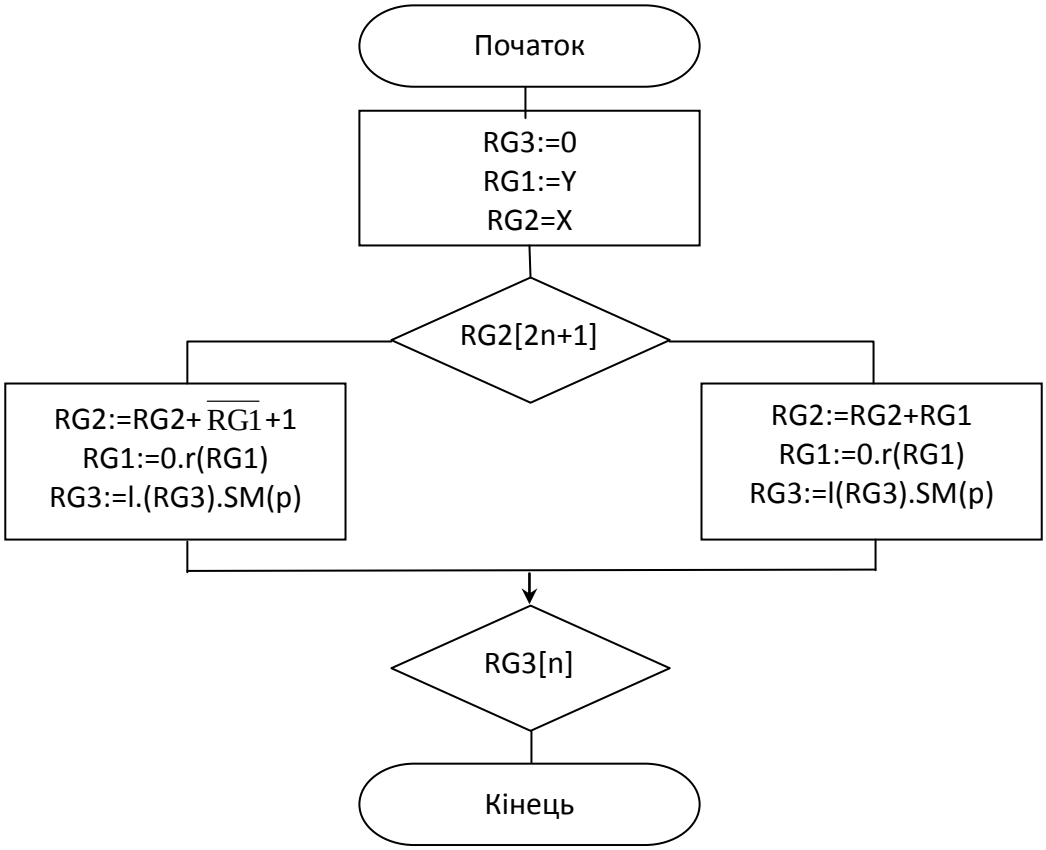


Рисунок 2.6.2-Змістовний мікроалгоритм

2.6.4 Таблиця станів регістрів

Таблиця 2.6.1- Таблиця станів регістрів

№	RG3 (Z)	RG2 (X)	RG1 (Y)
пс	0000000000000001	01011001110011100000000000000000	00010110011100110100000000000000
1	0000000000000011	01011001110011100000000000000000 + 11101001100011001100000000000000 01000011010110101100000000000000	00001011001110011010000000000000
2	0000000000000110	01000011010110101100000000000000 + 11110100110001100110000000000000 00111000001000010010000000000000	00000101100111001101000000000000
3	0000000000001101	00111000001000010010000000000000 + 11111010011000110011000000000000 00110010100001000101000000000000	00000010110011100110100000000000
4	0000000000011010	00110010100001000101000000000000 + 11111101001100011001100000000000 00101111101101011110100000000000	00000001011001110011010000000000
5	0000000000110100	00101111101101011110100000000000 +	00000000101100111001101000000000

		1111111010011000110011000000000 0010111001001110101101000000000	
6	0000000001101000	0010111001001110101101000000000 + 1111111101001100011001100000000 0010110110011011000110100000000	0000000001011001110011010000000
7	0000000011010000	0010110110011011000110100000000 + 1111111110100110001100110000000 0010110101000001010011010000000	0000000000101100111001101000000
8	0000000110100001	0010110101000001010011010000000 + 1111111111010011000110011000000 0010110100010100011001101000000	0000000000010110011100110100000
9	0000001101000010	0010110100010100011001101000000 + 1111111111101001100011001100000 0010110011111101111100110100000	0000000000001011001110011010000
10	0000011010000101	0010110011111101111100110100000 + 1111111111110100110001100110000 0010110011110010101110011010000	0000000000000101100111001101000
11	0000110100001010	0010110011110010101110011010000 + 1111111111111010011000110011000 0010110011101101000111001101000	0000000000000010110011100110100
12	0001101000010100	0010110011101101000111001101000 + 1111111111111101001100011001100 0010110011101010010011100110100	0000000000000001011001110011010
13	0011010000101000	0010110011101010010011100110100 + 1111111111111110100110001100110 0010110011101000111001110011010	0000000000000000101100111001101
14	0110100001010001	0010110011101000111001110011010 + 1111111111111111010011000110100 0010110011101000001100111001110	0000000000000000010110011100110
15	**1101000010100011**	0010110011101000001100111001110 + 1111111111111111101001100011010 0010110011100111111011001110100	0000000000000000001011001110011

2.6.5 Функціональна схема з відображенням управляючих сигналів

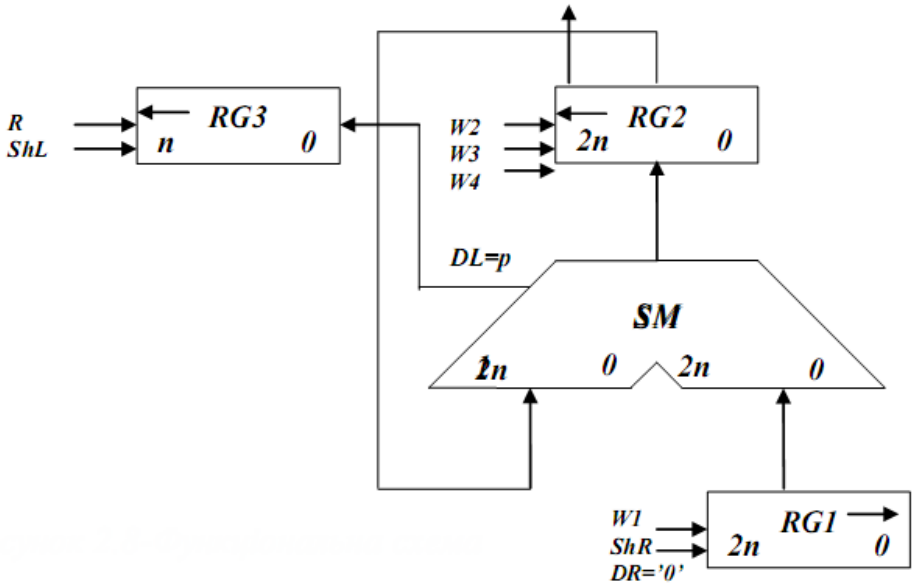


Рисунок 2.6.3-Функціональна схема

2.6.6 Закодований мікроалгоритм

Таблиця 2.6.2- Таблиця кодування мікрооперацій

Таблиця кодування мікрооперацій			Таблиця кодування логічних умов
МО	УС	ЛУ	Позначення
RG3:=0 RG1:=Y RG2:=X RG2:=RG2+RG1 RG1:=0.r(RG1) RG3:=l(RG3).SM(p) RG2:==RG2+$\overline{RG1}$+1	R W1 W2 W3 ShR ShL W4	RG2[2n+1] RG3[n]	X1 X2

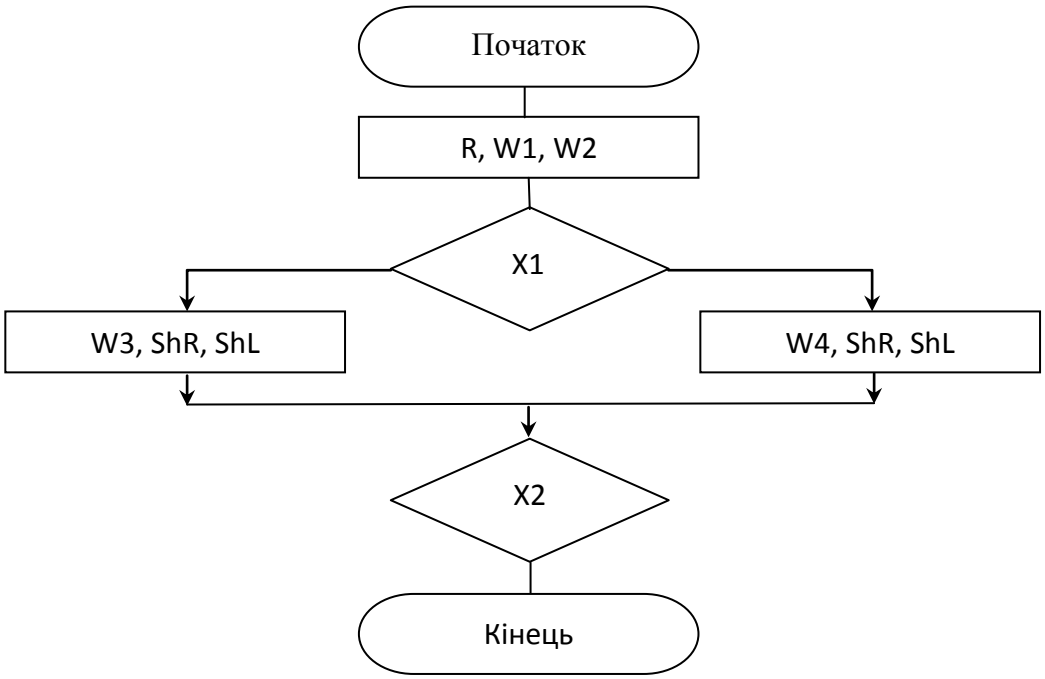


Рисунок 2.6.4- Закодований мікроалгоритм

2.6.7 Граф управляючого автомата Мура з кодами вершин

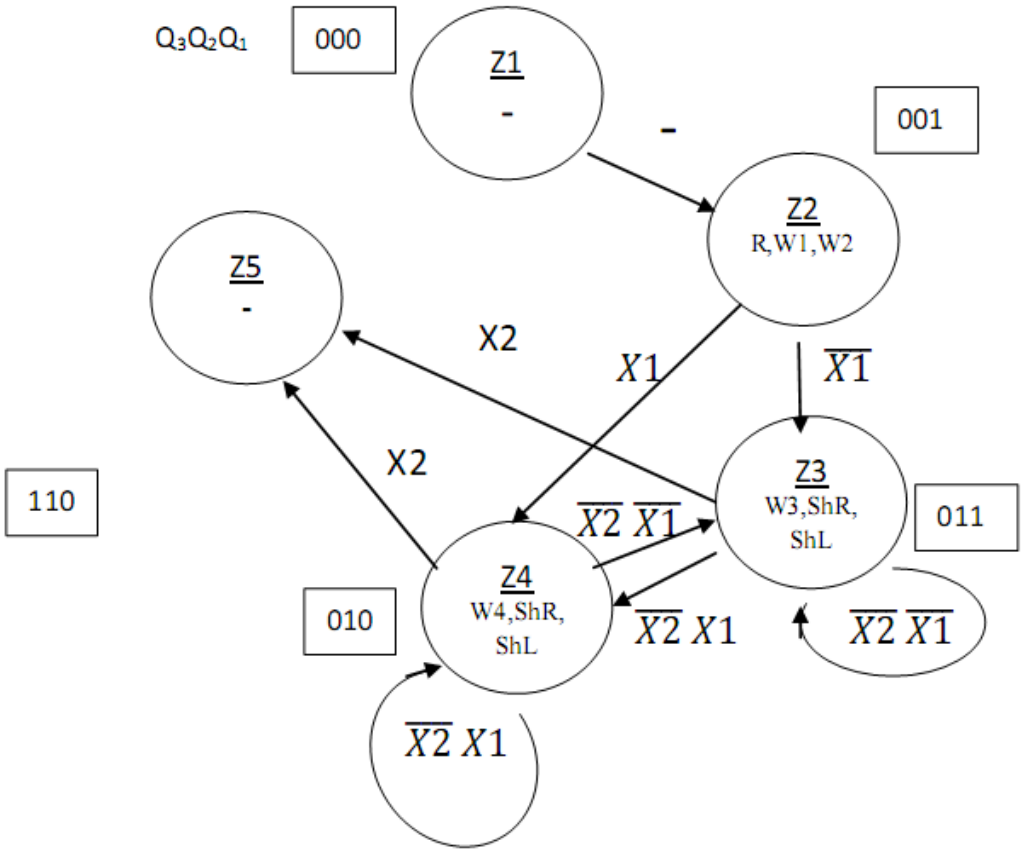


Рисунок 2.6.5- Граф автомата Мура

2.6.8 Обробка порядків:

Порядок частки буде дорівнювати: $P_z = P_x - P_y$;

В моєму випадку P_x=8; P_y=5; P_z=3;

2.6.8 Нормалізація результату:

Отримали результат: **1101000010100011**

Знак мантиси: 1 ⊕ 0 = 1.

Нормалізація мантиси не потрібна.

0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	1	0	1	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.

1. Порівняння порядків.

$P_x=+8_{10}=+1000_2$

$P_y=+5_{10}=+0101_2$

$P_x > P_y \Rightarrow$

$\Delta = P_x - P_y = 8_{10}-5_{10}=3_{10}=11_2$

2. Вирівнювання порядків.

Робимо зсув вправо мантиси числа Y, зменшуючи Δ на кожному кроці, доки Δ не стане 0.

Таблиця 2.7.1- Таблиця зсуву мантиси на етапі вирівнювання порядків

M_Y	Δ	Мікрооперація
0,101100111001101	11	Початковий стан
0,010110011100110	10	$M_y=0.r(M_y)$; Δ:=Δ-1
0,001011001110011	01	$M_y=0.r(M_y)$; Δ:=Δ-1
0,000101100111001	00	$M_y=0.r(M_y)$; Δ:=Δ-1

3. Додавання мантис у модифікованому ДК.

$X_{мдк}$ = 11. 010011000110011

$Y_{мдк}$ = 00. 000101100111001

Таблиця 2.7.2-Додавання мантис(для додавання)

M_X	1	1,	0	1	0	0	1	1	0	0	0	1	1	0	0	1	1
M_Y	0	0,	0	0	0	1	0	1	1	0	0	1	1	1	0	0	1
M_Z	1	1,	1	0	0	1	1	1	0	1	0	0	1	0	1	1	1

$Z_{дк}$ = 1. 100111010010111

$Z_{пк}$ = 1. 011000101101100

4. Нормалізація результату (В ПК).

$P_z = 7_{10} =111_2$

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

2.7.2 Операційна схема

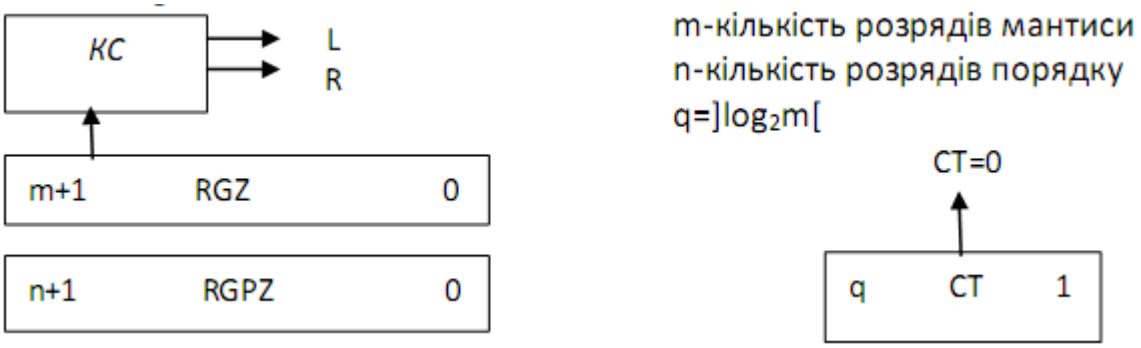


Рисунок 2.7.1-Операційна схема

Виконаємо синтез КС для визначення порушення нормалізації.

Таблиця 2.7.4-Визначення порушення нормалізації

Розряди регістру RGZ			Значення функцій	
Z'_0	Z_0	Z_1	L	R
0	0	0	0	1
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0

$L= Z_0$, $R=\overline{Z_1}$.

Результат беремо по модулю, знак встановлюємо за Z'_0 до нормалізації.

2.7.3 Змістовний алгоритм

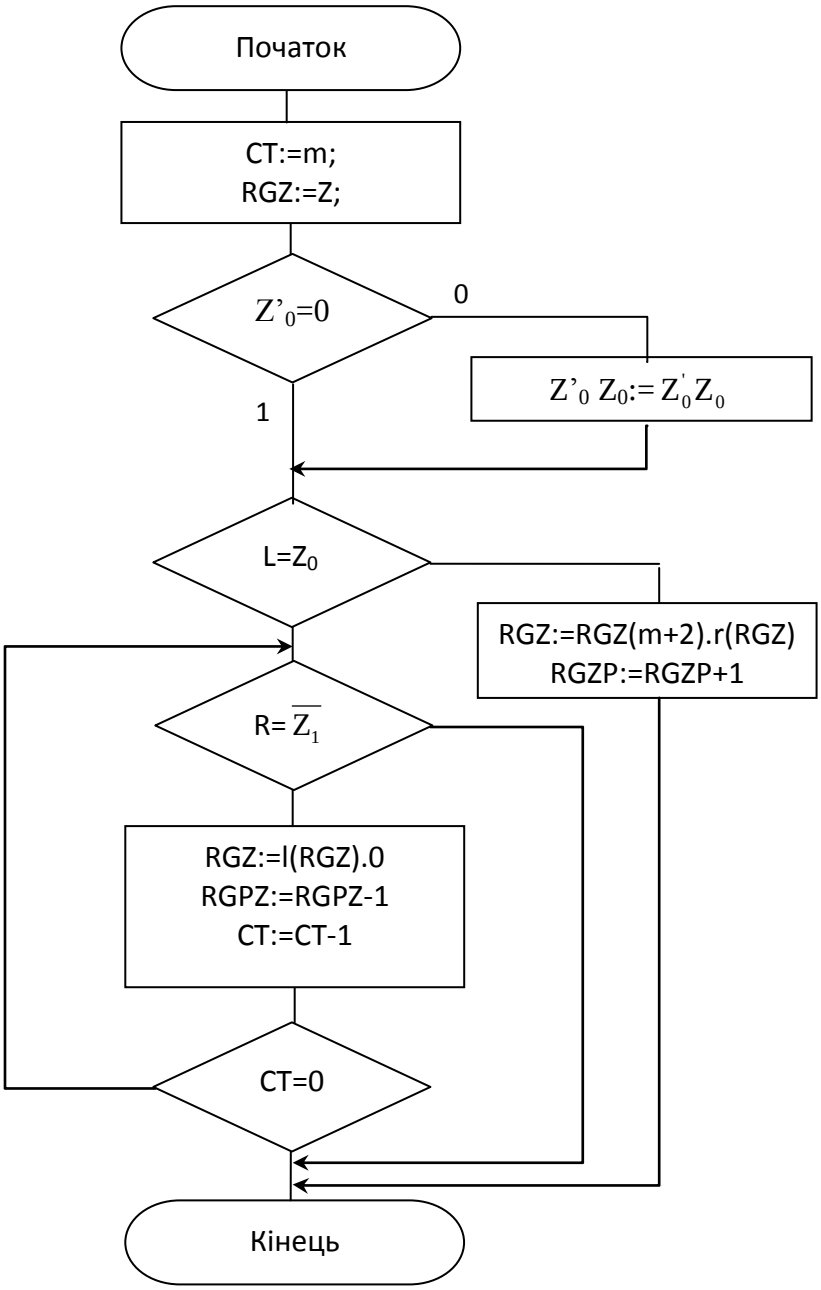


Рисунок 2.7.2-Змістовний мікроалгоритм

2.7.4 Таблиця станів регістрів

1) Додавання

Таблиця 2.7.5- Таблиця станів регістрів

№ такту	RGPZ	RGZ	ЛПН(L)	ППН(R)	СТ	Мікрооперація
ПС	001000	11. 100111010010111	0	1	100	

2.7.5 Функціональна схема з відображенням керуючих сигналів

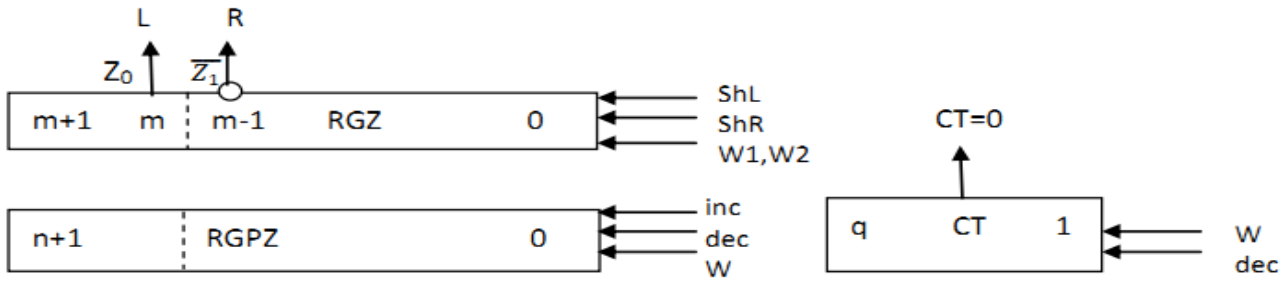


Рисунок 2.7.3 – Функціональна схема

2.7.6 Закодований мікроалгоритм

Таблиця 2.7.7– Таблиця кодування

Таблиця кодування мікрооперацій	
МО	УС
CT:=m;	W
RGZ:=Z;	W1
$Z'_0\ Z_0 := \overline{Z'_0}\ \overline{Z_0}$	W2
RGZ:=RGZ(m+2).r(RGZ)	ShR
RGPZ:=RGPZ+1	inc
RGZ:=l(RGZ).0	ShL
RGPZ:=RGPZ-1	dec
CT:=CT-1;	dec

Таблиця кодування логічних умов	
ЛУ	Позначення
$Z'_0=0$	X1
$L = Z_0$	X2
$R = \overline{Z_1}$	X3
$CT = 0$	X4

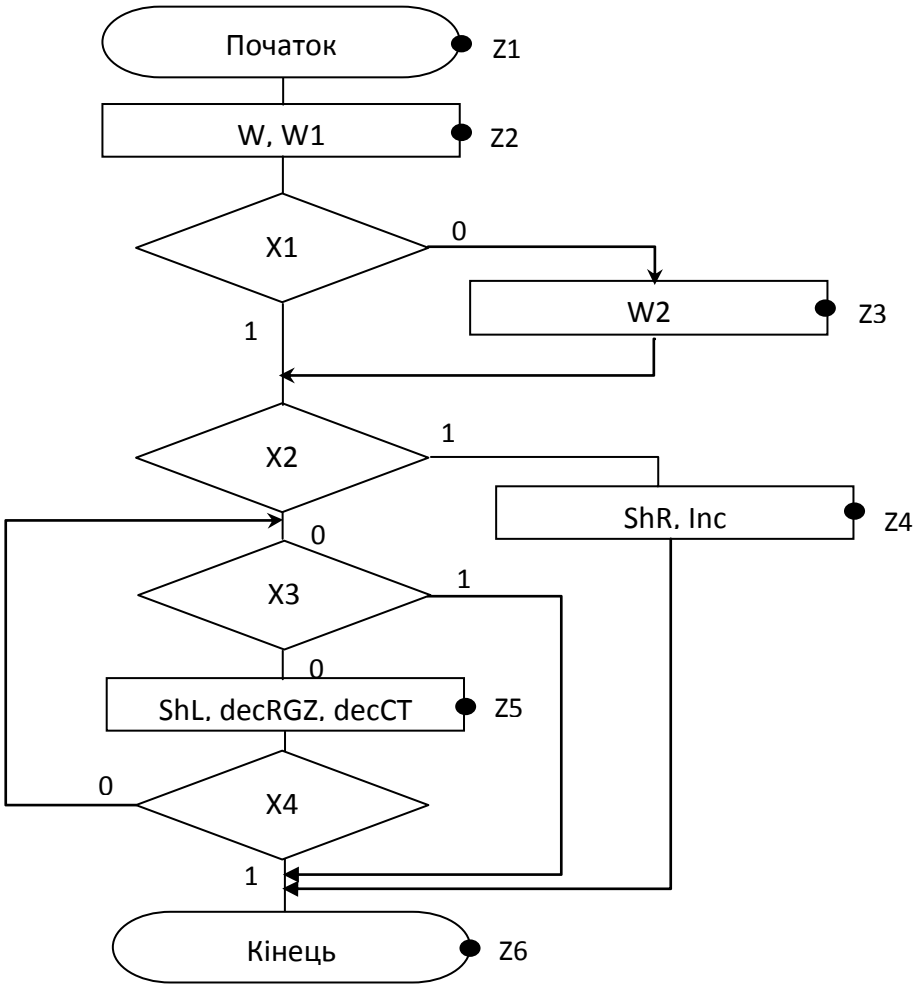


Рисунок 2.7.4 – Закодований мікроалгоритм

2.7.7 Граф управляючого автомата Мура з кодами вершин

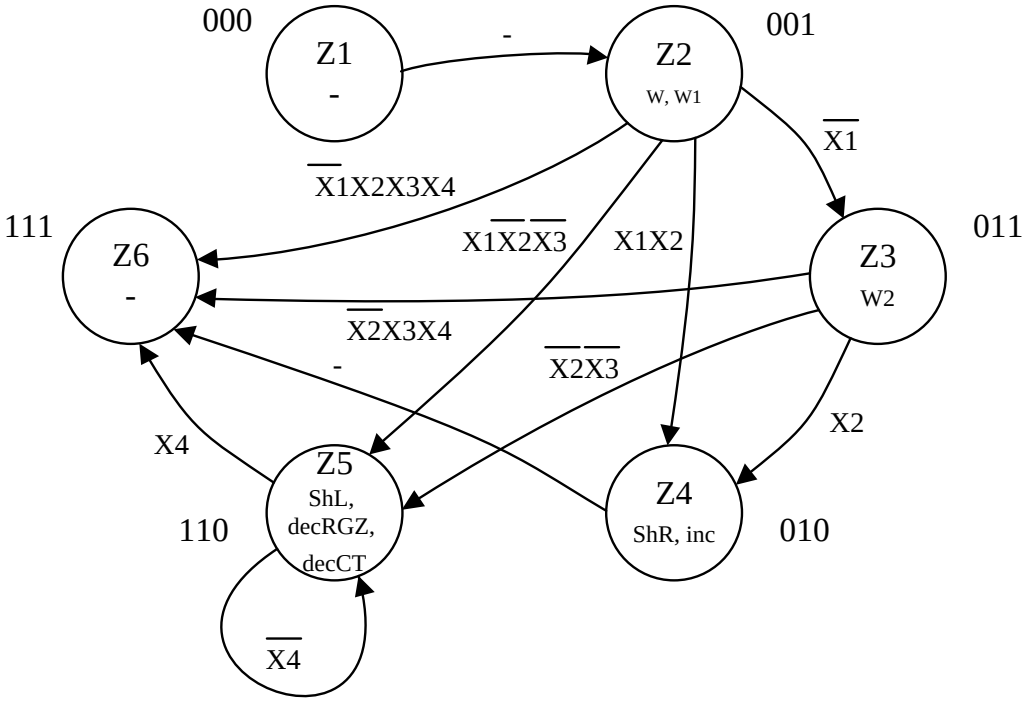


Рисунок 2.7.5 – Граф автомата Мура

2.7.8 Обробка порядків

$P_{X+Y}= 8_{10} =1000_2$

2.7.9 Форма запису результату з плаваючою комою

Результат додавання Z=X+Y.

$Z_{пк}$ = 1. 011000101101100
$P_z = 7_{10} =111_2$ M_z = - 110001011011000_2

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

2.8.Операція добування кореня

2.8.1Теоритичне обґрунтування операції обчислення квадратного кореня

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

$Z_i \le \sqrt{X} \le Z_i + 2^{-i}$;

$Z_i{}^2 \le X \le Z_i{}^2 + 2^{-i} Z_i + 2^{-2i}$;

$0 \le 2^{i-1}(X - Z_i{}^2) \le Z_i + 2^{-i-1}$.

Виконання операції зводиться до послідовності дій:

1. Одержання остачі.

$R_{i+1}{}' = 2R_i - Z_i - 2^{-i-2}$;

2. Якщо $R_{i+1}{}' \ge 0$, то $Z_{i+1} = 1$, $R_{i+1} = R_{i+1}{}'$.

3. Якщо $R_{i+1}{}' < 0$, то $Z_{i+1} = 0$, $R_{i+1} = R_{i+1}{}' + Z_i - 2^{-i-2}$.

Відновлення остачі додає зайвий такт, але можна зробити інакше:

$R_{i+2} = 2R_{i+1}{}' + Z_i + 2^{-i-2} + 2^{-i-3}$, тоді корінь добувається без відновлення залишку.

Для цього R_i зсувається на 2 розряди ліворуч, а Z_i - на 1 розряд ліворуч, і формується як при діленні.

2.8.2 Операційна схема операції обчислення квадратного кореня

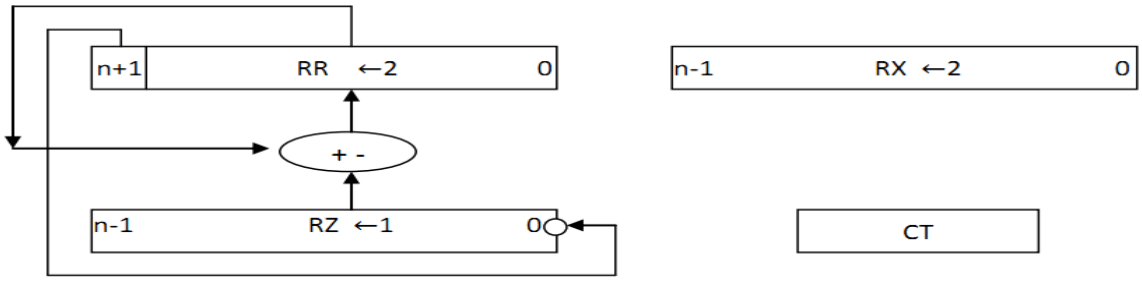


Рисунок 2.8.1 –Операційна схема

2.8.3 Змістовний мікроалгоритм

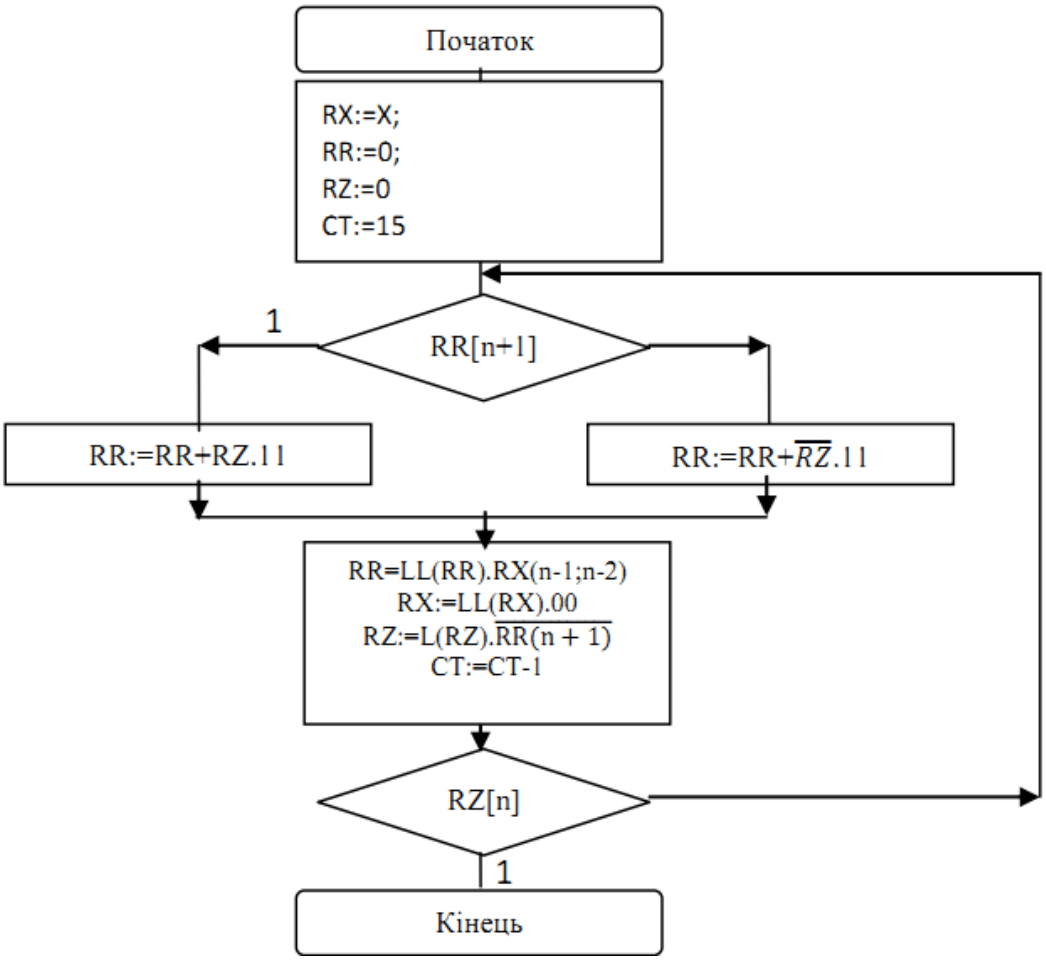


Рисунок 2.8.2 – Змістовний мікроалгоритм

2.8.4 Таблиця станів регістрів

Таблиця 2.8.1 – Таблиця станів регістрів

№	RZ	RR	RX	CT
пс		0000000000000000		
пз	000000000000000	0000000000000010	101100111001110	1111
1	000000000000001	0000000000000010 + 1111111111111111 0000000000000001 0000000000000111	010110011100111	1110
2	000000000000011	0000000000000111 + 1111111111111011 0000000000000010 0000000000001000	001011001110011	1101
3	000000000000110	0000000000001000 + 1111111111110011 1111111111111011 1111111111101111	000101100111001	1100

4	000000000001101	11111111111101111 + 00000000000011011 00000000000001010 00000000000101010	000010110011100	1011
5	000000000011010	00000000000101010 + 11111111111001011 11111111111110101 11111111111010101	000001011001110	1010
6	000000000110101	11111111111010101 + 00000000001101011 00000000001000000 00000000100000011	000000101100111	1001
7	000000001101011	00000000100000011 + 11111111100101011 00000000000101110 00000000010111000	000000010110011	1000
8	000000011010110	00000000010111000 + 11111111001010011 11111111100001011 11111110000101100	000000001011001	0111
9	000000110101100	11111110000101100 + 00000001101011011 11111111110000111 11111111000011100	000000000101100	0110
10	000001101011001	11111111000011100 + 00000011010110011 00000010011001111 00001001100111100	000000000010110	0101
11	000011010110011	00001001100111100 + 11111001010011011 00000010111010111 00001011101011100	000000000001011	0100
12	000110101100110	00001011101011100 + 11110010100110011 11111110010001111 11111001000111100	000000000000101	0011
13	001101011001101	11111001000111100	000000000000010	0010

		+ 00011010110011011 00010011111010111 01001111101011100		
14	011010110011011	01001111101011100 + 11001010011001011 00011010000100111 01101000010011100	000000000000001	0001
15	**110101100110110**	01101000010011100 + 10010100110010011 11111101000101111 11110100010111100	000000000000000	0000

2.8.5 Функціональна схема операції обчислення квадратного кореня

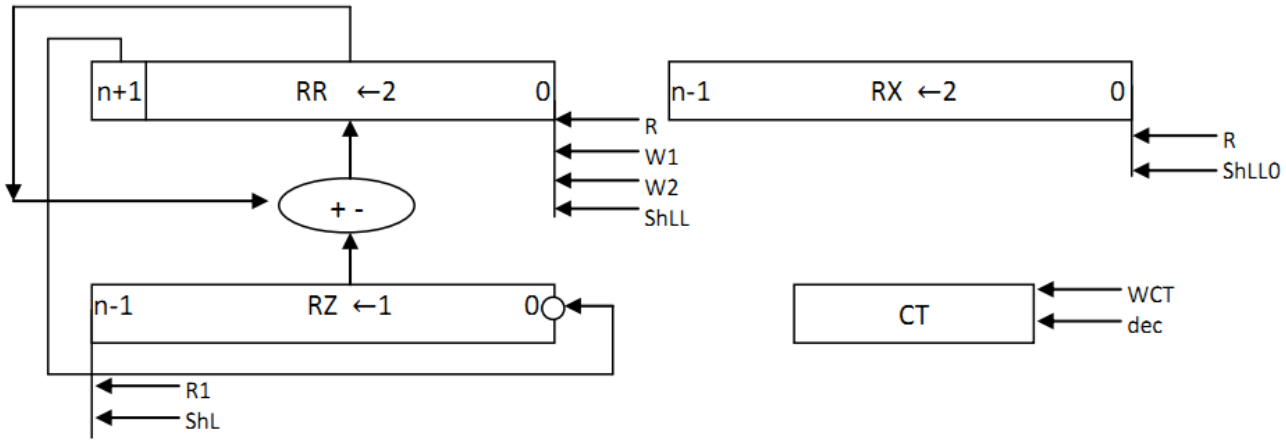


Рисунок 2.8.3 – Функціональна схема

2.8.6 Закодований мікроалгоритм

Таблиця 2.8.2 – Таблиця кодування

Таблиця кодування мікрооперацій	
МО	УС
RX:=X;	WX
RR:=0;	R
RZ:=0	R1
CT:=15	WCT
RR:=RR+RZ.11	W1
RR:=RR+$\overline{RZ}$.11	W2
RR=LL(RR).RX(n-1;n-2)	ShLL
RX:=LL(RX).00	ShLL0
RZ:=L(RZ).$\overline{RR(n+1)}$	ShL
CT:=CT-1	dec

Таблиця кодування логічних умов	
ЛУ	Позначення
RR[n+1]	X1
RZ[n]	X2

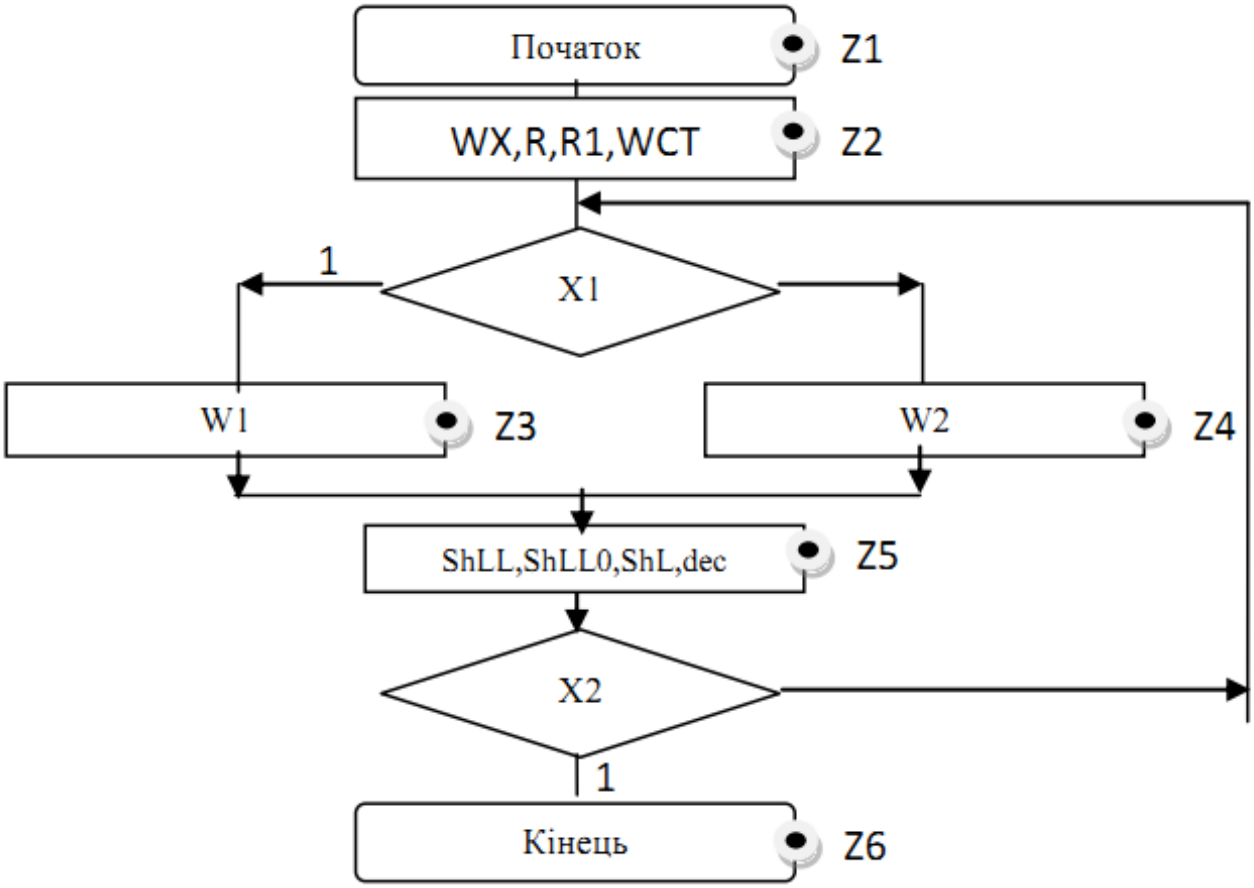


Рисунок 2.8.4 – Закодований мікроалгоритм

2.8.7Граф управляючого автомата Мура з кодами вершин

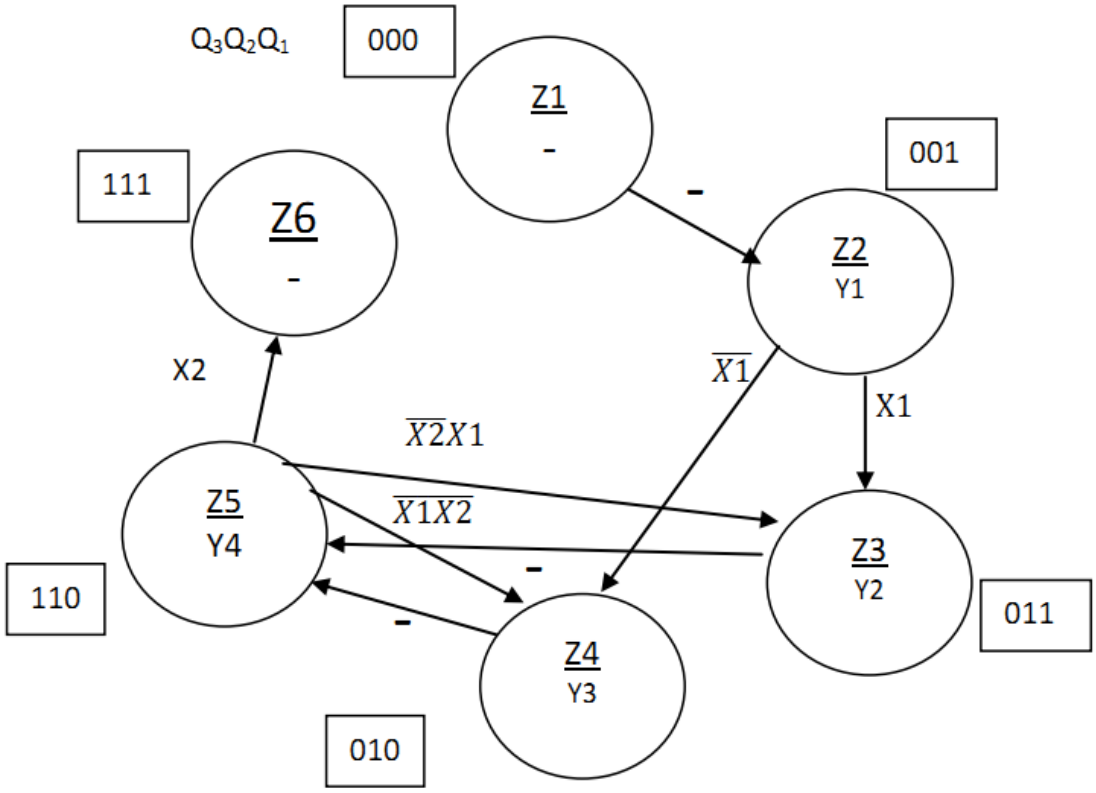


Рисунок 2.8.5 – Граф управляючого автомата Мура

2.8.8 Обробка порядків

$$P_z = P_x/2;$$

В моєму випадку P_z=4;

2.8.9 Запис результату

Отримали результат Z = **110101100110110**;

Результат нормалізований, готовий до запису у мантису:

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

Завдання 3

$x_3x_2x_1$ =6– операція додавання.

3.1 Таблиця співвідношення управляючих входів операційного автомата і виходів управляючого автомата

За закодованим мікроалгоритмом складемо таблицю:

Таблиця 3.1 Таблиця кодування сигналів

Входи операційного автомата	Виходи управляючого автомата
CLR1, W2, W3, W_{CT}	Y1
W1	Y2
SR1,SR2,DEC	Y3

3.2 Мікроалгоритм в термінах управляючого автомата

Зробимо автомат Мура циклічним задля зменшення кількості вершин.

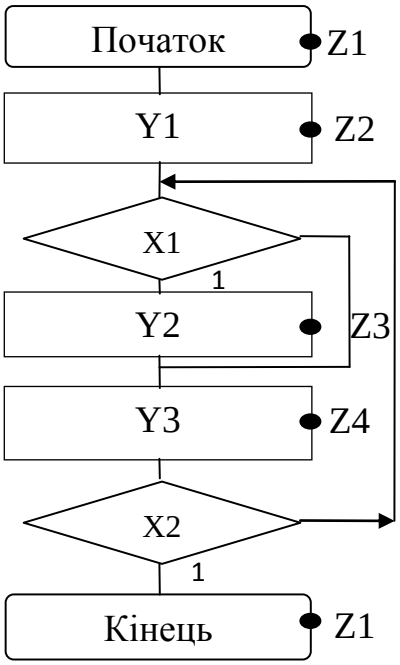


Рисунок 3.1- Закодований мікроалгоритм

Будуємо граф автомата Мура

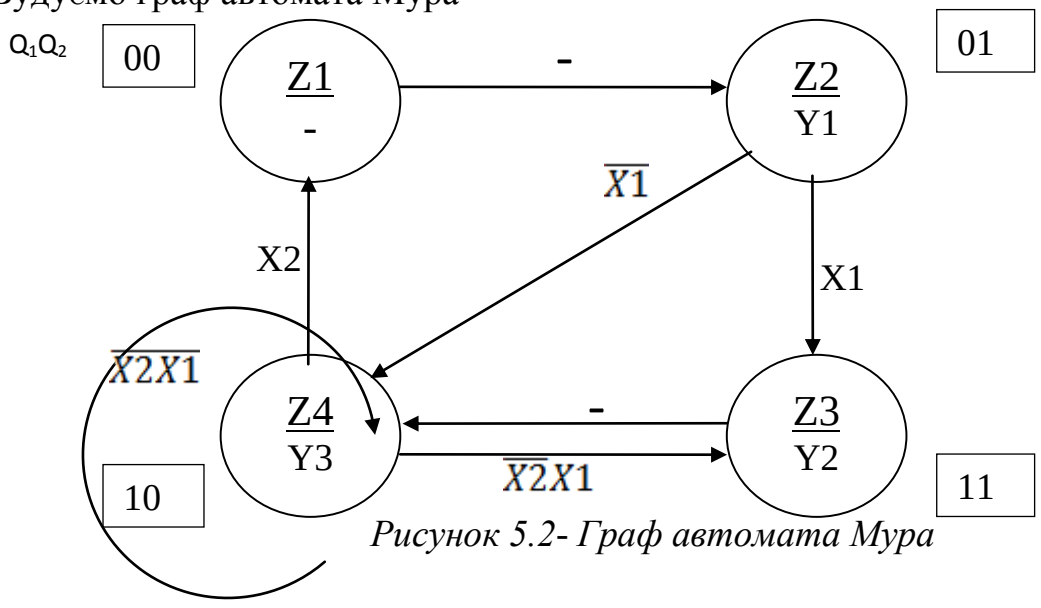


Рисунок 5.2- Граф автомата Мура

3.3 Структурна таблиця автомата

3.3.1Таблиця співвідношення управляючих входів операційного автомата і виходів управляючого автомата

Таблиця 3.1 - Таблиця співвідношення управляючих входів операційного автомата і виходів управляючого автомата

управляючі входи	виходи управляючого автомата
W_M, W_P	Y1
SHR_M, inc_P	Y2
W_{CT}	Y3
SHL_M, inc_P, dec_{CT}	Y4

3.3.2Мікроалгоритм в термінах управляючого автомата

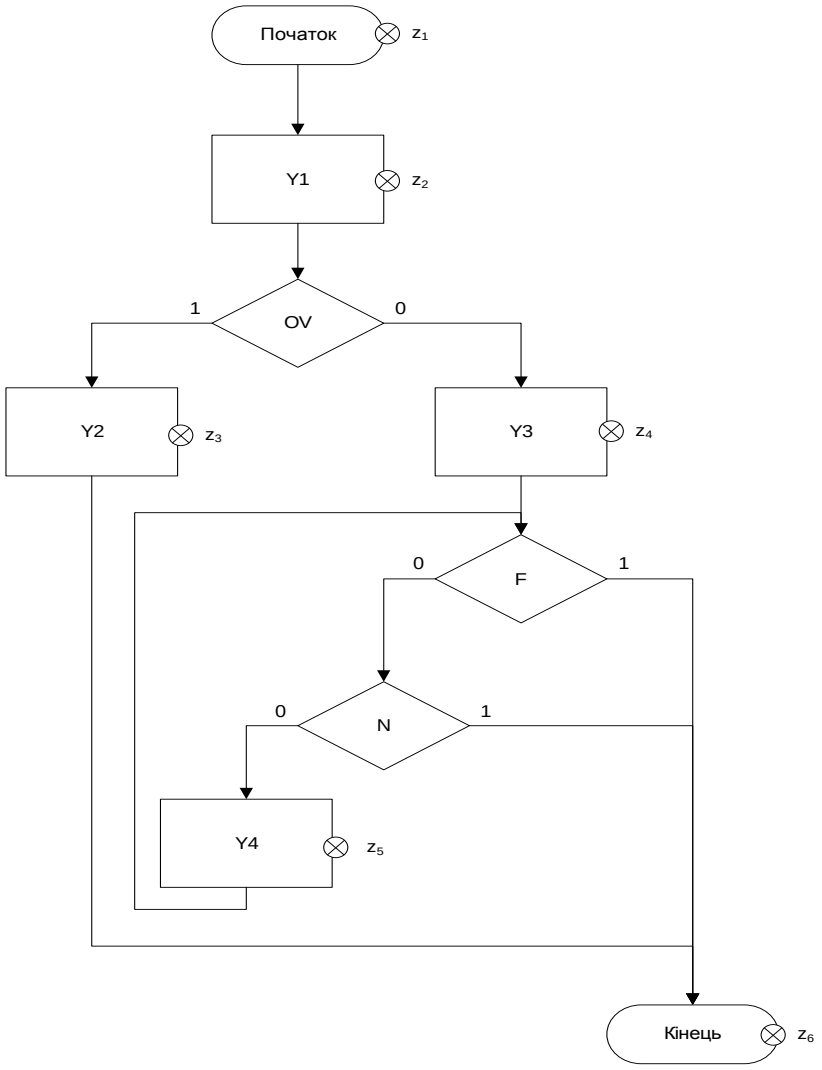


Рисунок 3.1 – Закодований мікроалгоритм додавання

3.3.3Структурна таблиця автомата

Таблиця 3.2 - Структурна таблиця автомата

$Q_3Q_2Q_1$	$Q_3Q_2Q_1$	OV F N	$Y_1Y_2Y_3Y_4$	J_3K_3	J_2K_2	J_1K_1
0 0 0	0 0 1	- - -	0 0 0 0	0 -	0 -	1 -
0 0 1	0 1 1	1 - -	1 0 0 0	0 -	1 -	- 0
0 0 1	1 0 1	0 - -	1 0 0 0	1 -	0 -	- 0
0 1 1	1 1 1	- - -	0 1 0 0	1 -	- 0	- 0
1 0 1	1 1 0	- 0 0	0 0 1 0	- 0	1 -	- 1
1 0 1	1 1 1	- 1 1	0 0 1 0	- 0	1 -	- 0
1 1 0	1 1 0	- 0 0	0 0 0 1	- 0	- 0	0 -
1 1 0	1 1 1	- 1 1	0 0 0 1	- 0	- 0	1 -

3.4Синтез функцій виходів і переходів

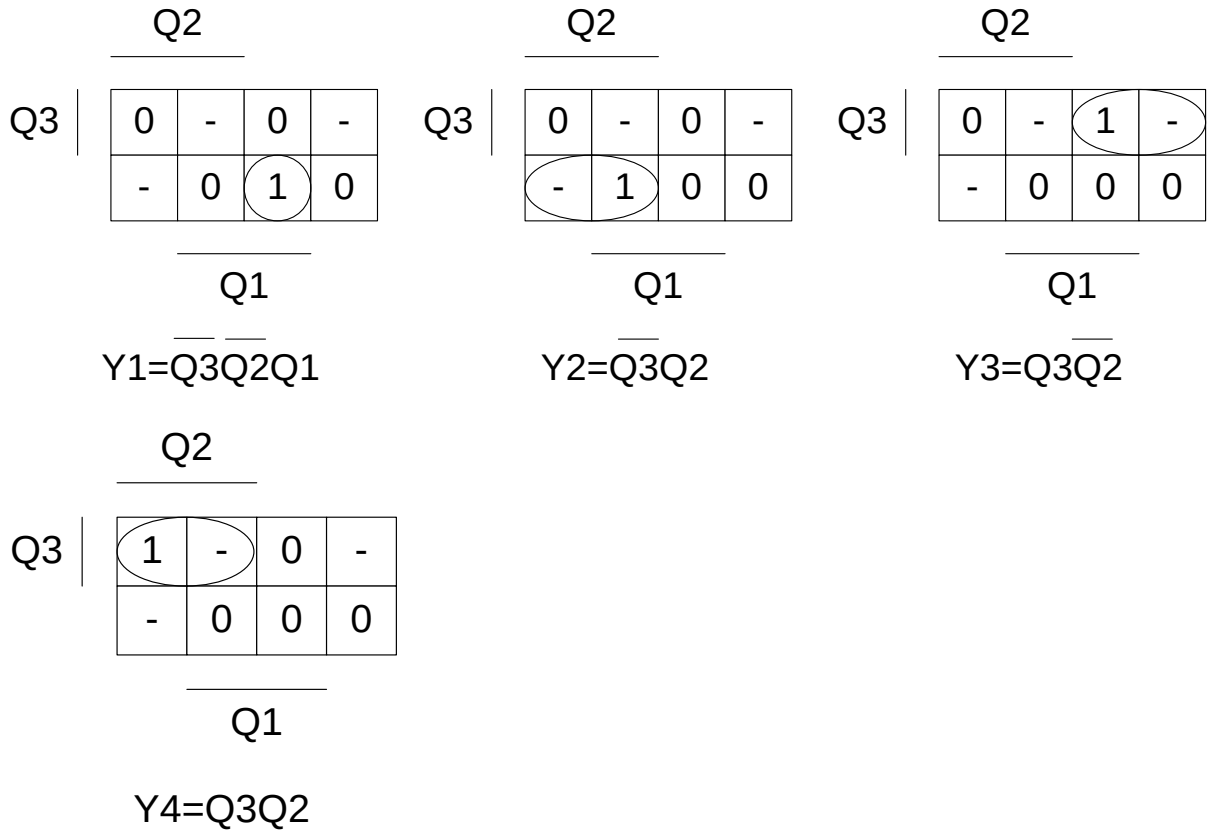
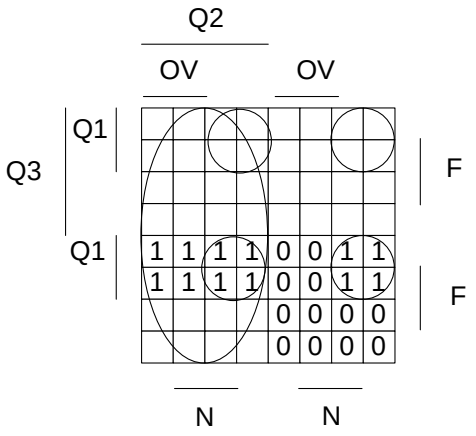
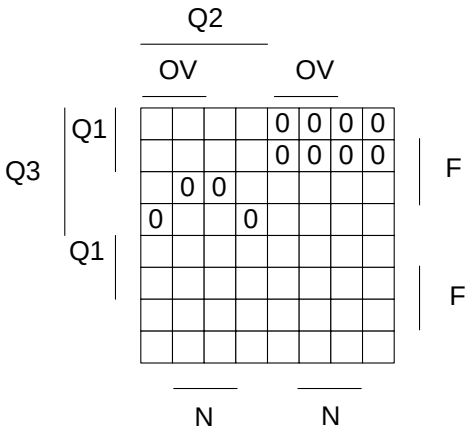


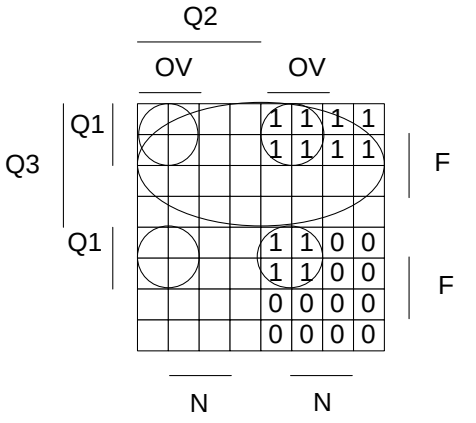
Рисунок 3.2 – Діаграми Вейча для функцій виходу



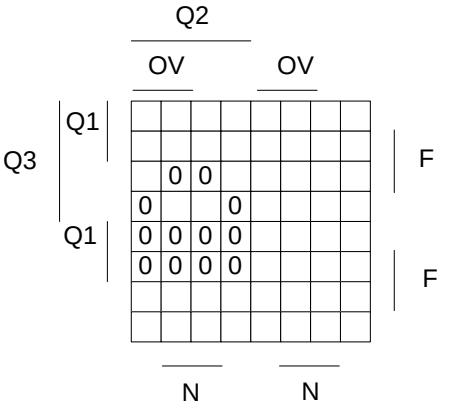
$J3=Q2 \vee Q1\overline{OV}$



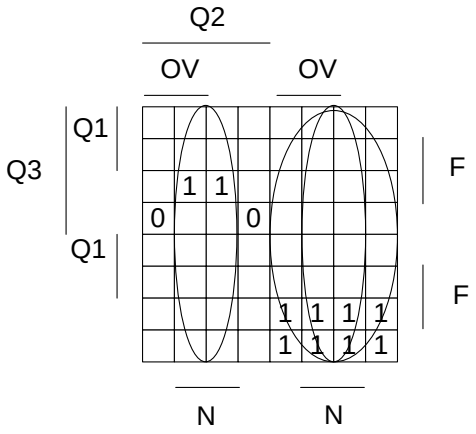
$K3=0$



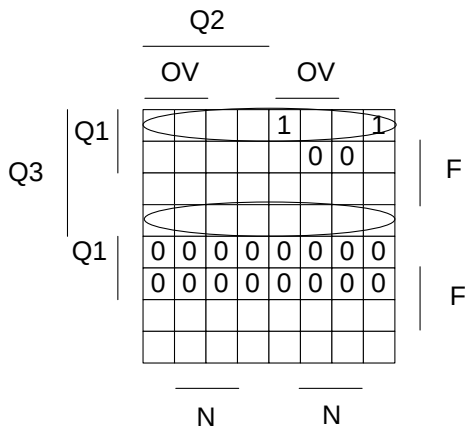
$J2=Q3 \vee Q1OV$



$K2=0$



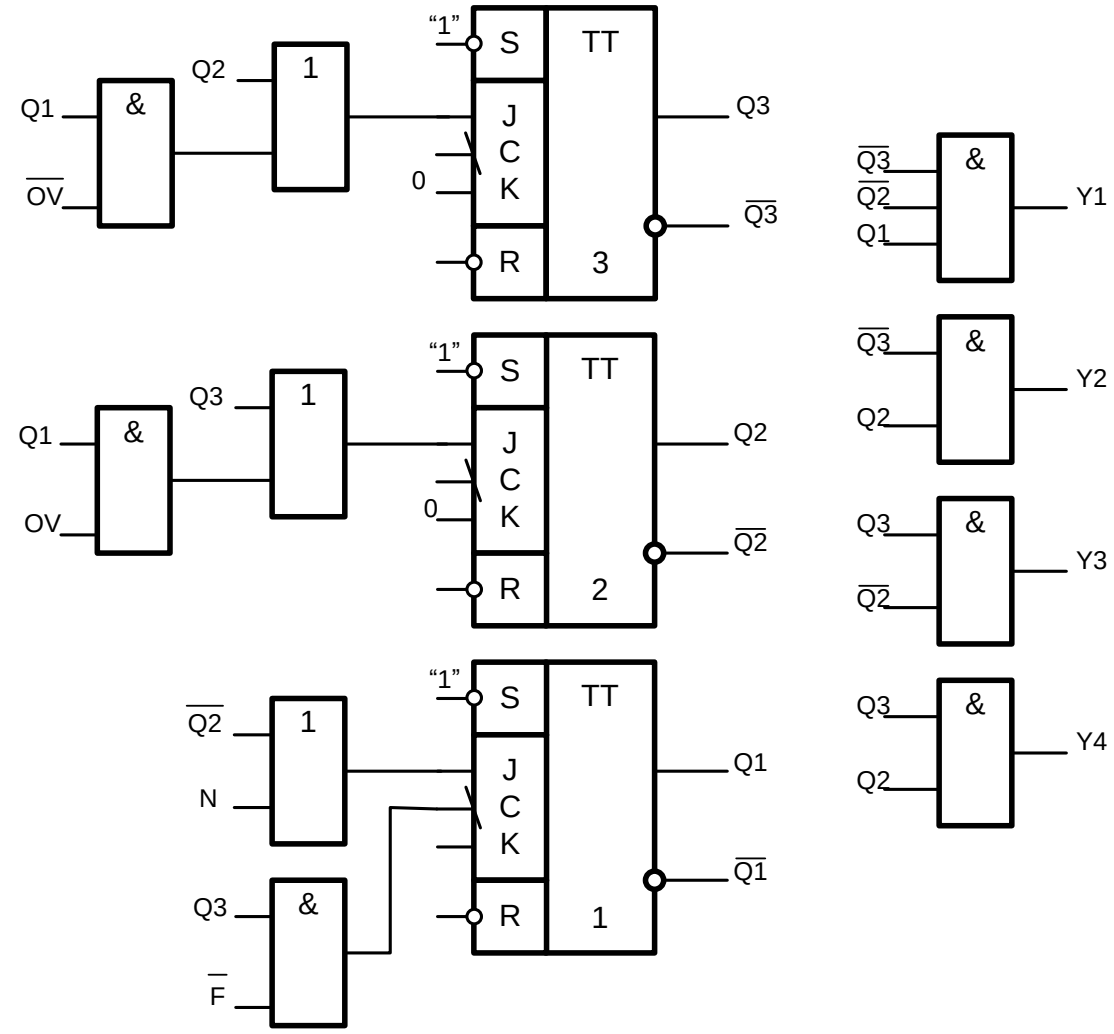
$J1=\overline{Q2} \vee N$



$K1=Q3\overline{F}$

Рисунок 3.3 – Діаграми Вейча для функцій переходу

3.5 Функціональна схема пристою (виходи управляючого автомата підключені до входів операційного автомата)



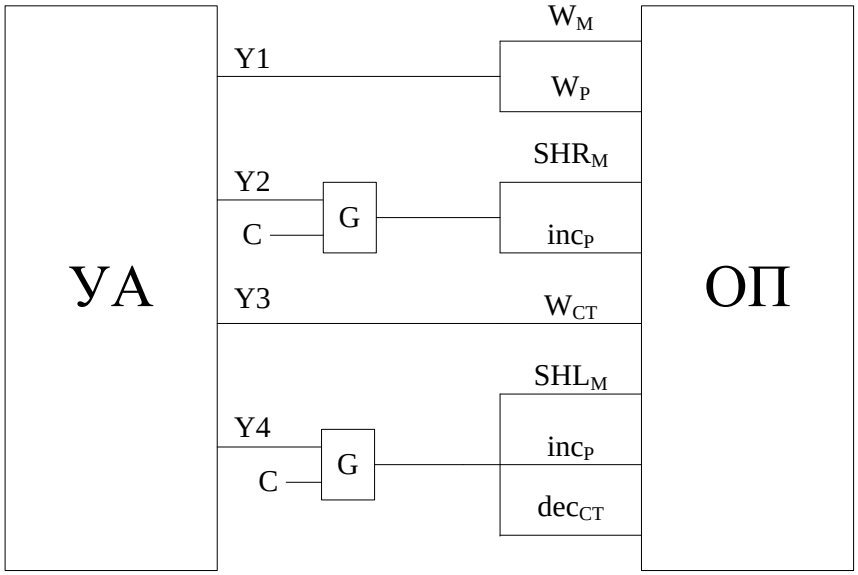


Рисунок 3.4 – Функціональна схема управляючого автомата

Висновок

У даній розрахунковій роботі було виконано операції з числами в двійковому коді з плаваючою комою, а саме: множення чотирма способами, ділення двома способами та додавання.

Було побудувано управляючий автомат Мура на тригерах JK і елементах булевого базиса для операції додавання.

Зроблено мінімізацію функцій тригерів і в середовищі AFDK побудована функціональна схема автомата.

Під час виконання даної розрахункової роботи я повторив для себе матеріал курсу «Компютерна логіка - 1», а також закріпив знання з курсу «Компютерна логіка - 2».

Було використано наступну літературу:

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.

2) Конспект лекцій з курсу «Комп'ютерна логіка - 1»

3) Конспект лекцій з курсу «Комп'ютерна логіка - 2»