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

РОЗРАХУНКОВА РОБОТА

по курсу „Комп'ютерна логіка-2”

Виконав: Бас Андрій Васильович
Група ІО-22, Факультет ІОТ,
Залікова книжка № 2201

Номер технічного завдання 100010011001

_____

(підпис керівника)

Київ – 2013 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 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 додавання та віднімання). Номери операцій (для п.3) відповідають порядку переліку (наприклад, 6 – ділення другим способом). 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 (x_2x_1 =00 відповідає RS-тригеру; 01 – D-тригеру; 10 – JK-тригеру; 11 – T-тригеру) 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).

$2201_{10}=100010011001_2$;

$X_2 = -1x_{10}x_9 1x_8x_7x_6 1, x_5x_4 0x_3 1x_2x_1$ = - 10011001,1100101;

$Y_2 = +1x_{10}1x_9x_8, x_7x_6 1x_5x_4 0x_3x_2x_1 1$ = + 10101,0011100011;

Основна частина:

Завдання №1

$X_{пк}$=1. 10011001,1100101;

$Y_{пк}$=0. 10101,0011100011;

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

X_2:

0	0	0	0	1	0	0	0

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

Y_2:

0	0	0	0	0	1	0	1

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

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

$E = P + 2^m$,

m = 7;

$2^7 = 10000000_2$

$E_x = 10000000 + 1000 = 10001000$

X_2:

1	0	0	0	1	0	0	0

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

$E_y = 10000000 + 101 = 10000101$

Y_2:

1	0	0	0	0	1	0	1

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

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:

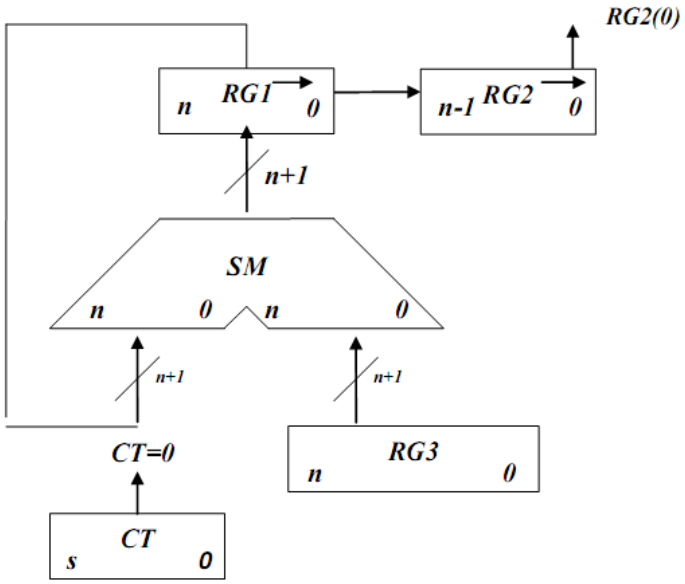

Рисунок 2.1.1- Операційна схема.

2.1.3 Змістовний мікроалгоритм:

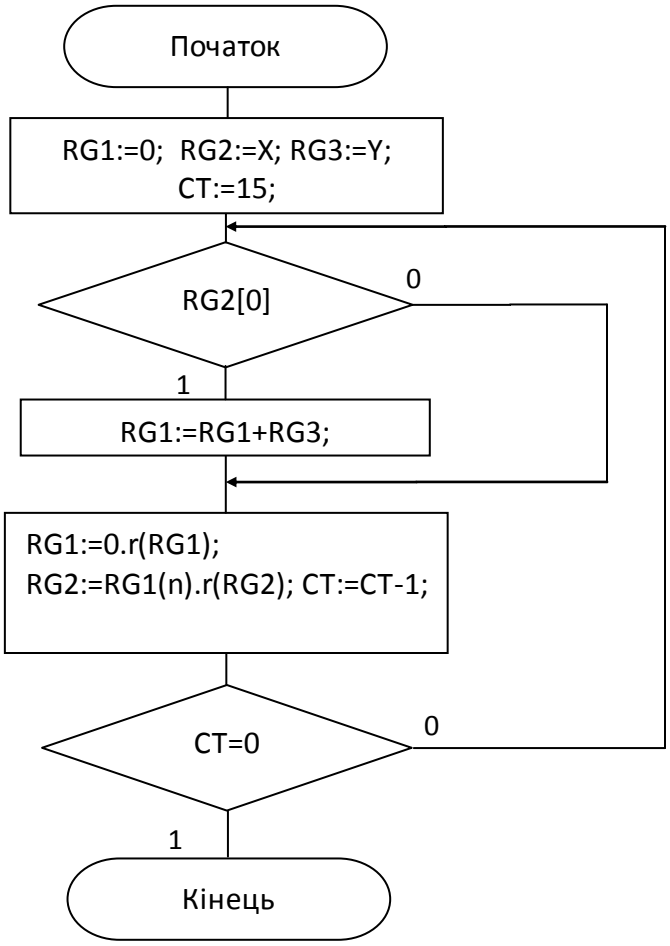

Рисунок 2.1.2 - Змістовний мікроалгоритм виконання операції множення першим способом.

2.1.4 Таблиця станів регістрів:

Таблиця 2.1.1-Таблиця станів регістрів для першого способу множення.

№	RG1	RG2	RG3	CT
пс	0	100110011100101	0101010011100011	1111
1	+0101010011100011 =0101010011100011 0010101001110001	110011001110010		1110
2	0001010100111000	111001100111001		1101
3	+0101010011100011 =0110101000011011 0011010100001101	111100110011100		1100
4	0001101010000110	111110011001110		1011
5	0000110101000011	011111001100111		1010
6	+0101010011100011 =0110001000100110 0011000100010011	001111100110011		1001
7	+0101010011100011 =1000010111110110 0100001011111011	000111110011001		1000
8	+0101010011100011 =1001011111011110 0100101111101111	000011111001100		0111
9	0010010111110111	100001111100110		0110
10	0001001011111011	110000111110011		0101
11	+0101010011100011 =0110011111011110 0011001111101111	011000011111001		0100
12	+0101010011100011 =1000100011010010 0100010001101001	001100001111100		0011
13	0010001000110100	100110000111110		0010
14	0001000100011010	010011000011111		0001
15	+0101010011100011 =0110010111111101 **0011001011111110**	**101001100001111**		0000

2.1.5 Функціональна схема:

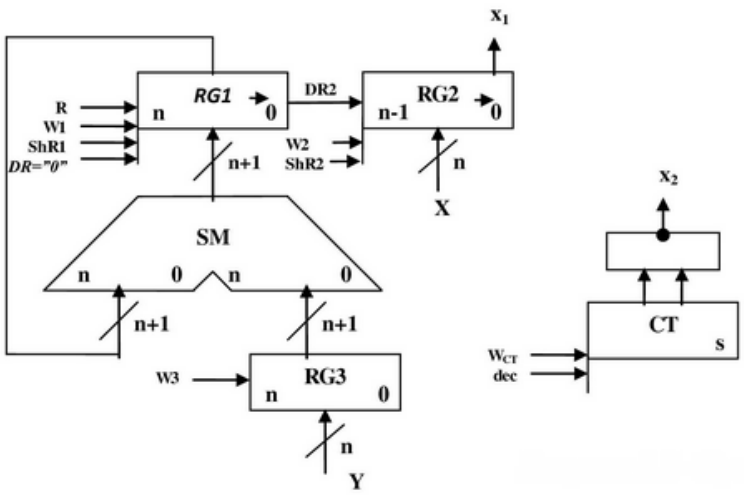

Рисунок 2.1.3- Функціональна схема.

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		

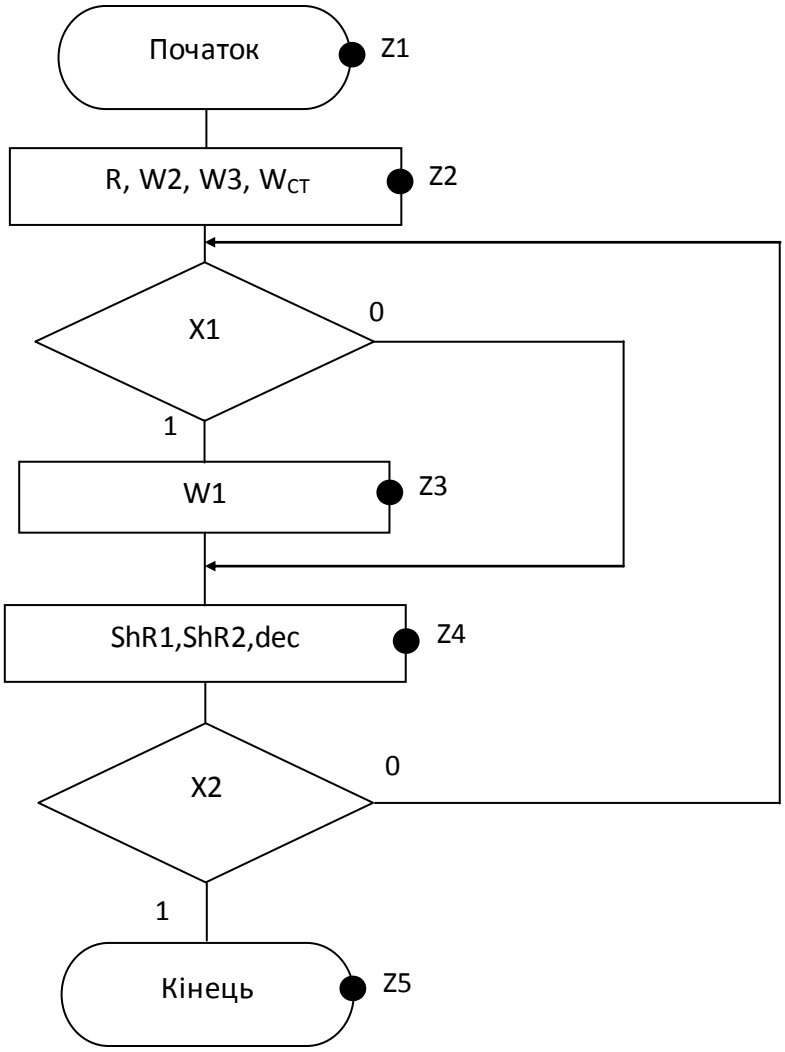

Рисунок 2.1.4-Закодований мікроалгоритм.

2.1.7 Граф управляючого автомата Мура з кодами вершин:

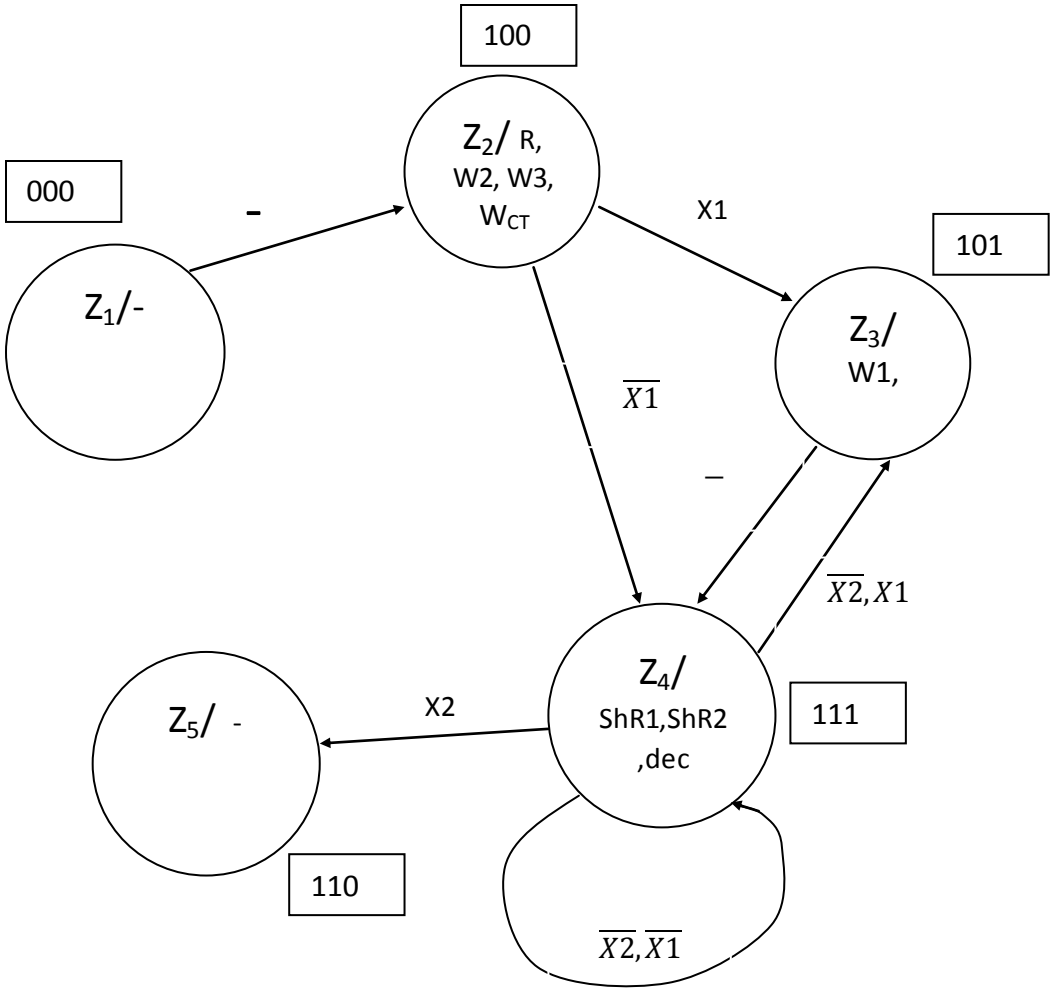

Рисунок 2.1.5-Граф автомата Мура

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: `01100101111111010100110000111 1`

Знак мантиси: 1 ⊕ 0 = 1.

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

Порядок зменшуємо на 1:

`11001011111110101001100001111`; P_z=12;

Запишемо нормалізований результат:

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

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

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

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

$$Z_0 = 0;$$

$$Y_0 = 0$$

2.2.2 Операційна схема:

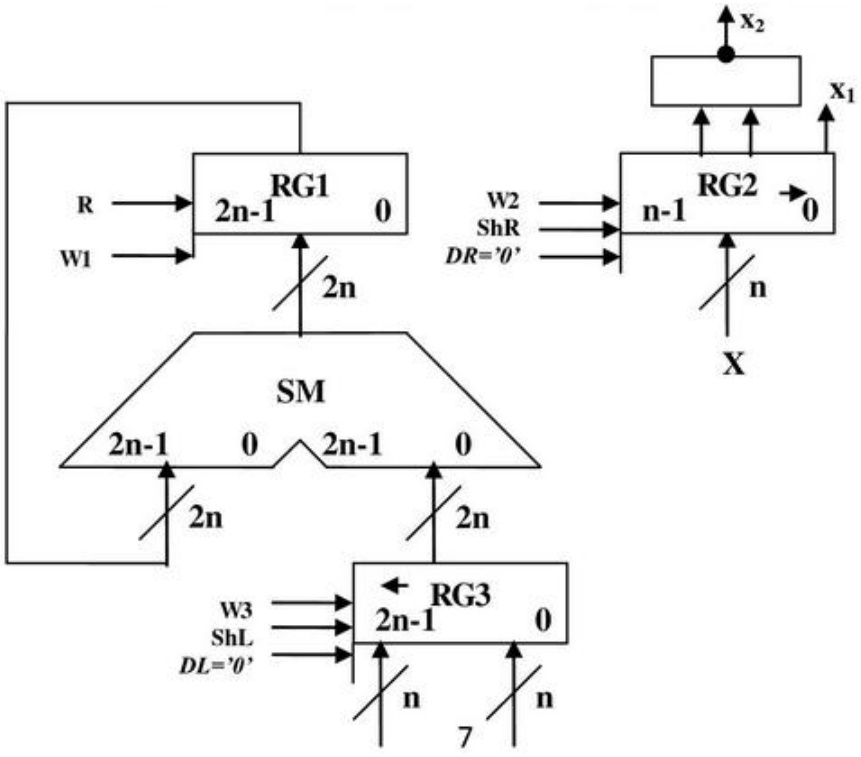

Рисунок 2.2.1- Операційна схема

2.2.3 Змістовний мікроалгоритм:

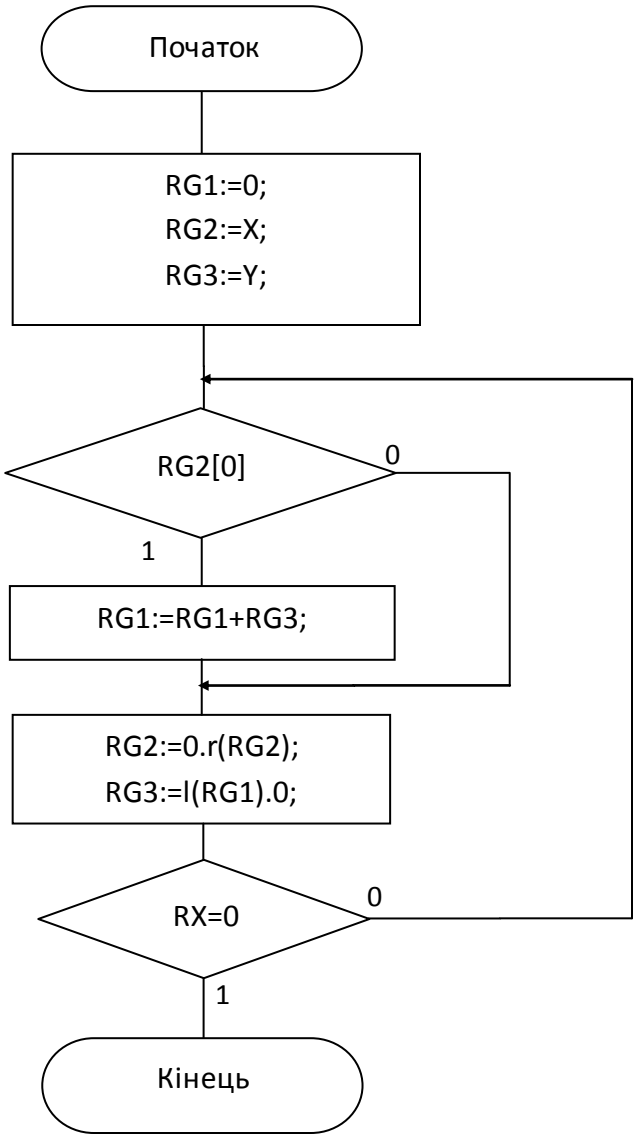

Рисунок 2.2.2 - Змістовний мікроалгоритм.

2.2.4 Таблиця станів регістрів:

Таблиця 2.2.1-Таблиця станів регістрів.

№	RG1	RG3 ←	RG2 →
пс	0	00000000000000001010010011100011	100110011100101
1	+00000000000000001010010011100011 =00000000000000001010010011100011	00000000000000001010100111000110	010011001110010
2	00000000000000001010010011100011	00000000000001010100111000110 0	001001100111001
3	+00000000000001010100111000110 0 =00000000000001101010000110 1111	00000000000010101001110001100 0	000100110011100
4	00000000000001101010000110 1111	00000000001010100111000110 0000	000010011001110
5	00000000000001101010000110 1111	00000000010101001110001100 0000	000001001100111

6	+00000000001010100111000110000000 =00000000001100010001001100111 1	00000000010101001110001100000 0	000000100110011
7	+00000000010101001110001100000 0 =00000000100001011111011000111 1	00000000101010011100011000000 0	000000010011001
8	+00000000101010011100011000000 0 =00000001001011111011110000111 1	00000001010100111000110000000 0	000000001001100
9	00000001001011111011110000111 1	00000010101001110001100000000 0	000000000100110
10	00000001001011111011110000111 1	00000101010011100011000000000 0	000000000010011
11	+00000101010011100011000000000 0 =00000110011111011110110000111 1	00001010100111000110000000000 0	000000000001001
12	+00001010100111000110000000000 0 =00010001000110100100110000111 1	00010101001110001100000000000 0	000000000000100
13	00010001000110100100110000111 1	00101010011100011000000000000 0	000000000000010
14	00010001000110100100110000111 1	01010100111000110000000000000 0	000000000000001
15	+01010100111000110000000000000 0 =**01100101111111010100110000111 1**	10101001110001100000000000000 0	000000000000000

2.2.5 Функціональна схема:

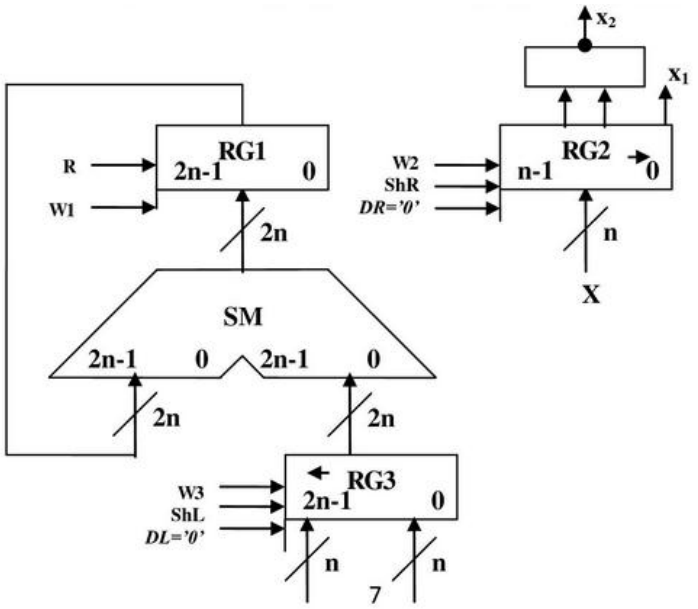

Рисунок 2.2.3- Функціональна схема.

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	R	RG2[0]	X1
RG2:=X	W2	RG2=0	X2
RG3:=Y	W3		
RG1:=RG1+RG3	W1		
RG2:=0.r(PG2)	ShR		
RG3:=l(RG3).0	ShL		

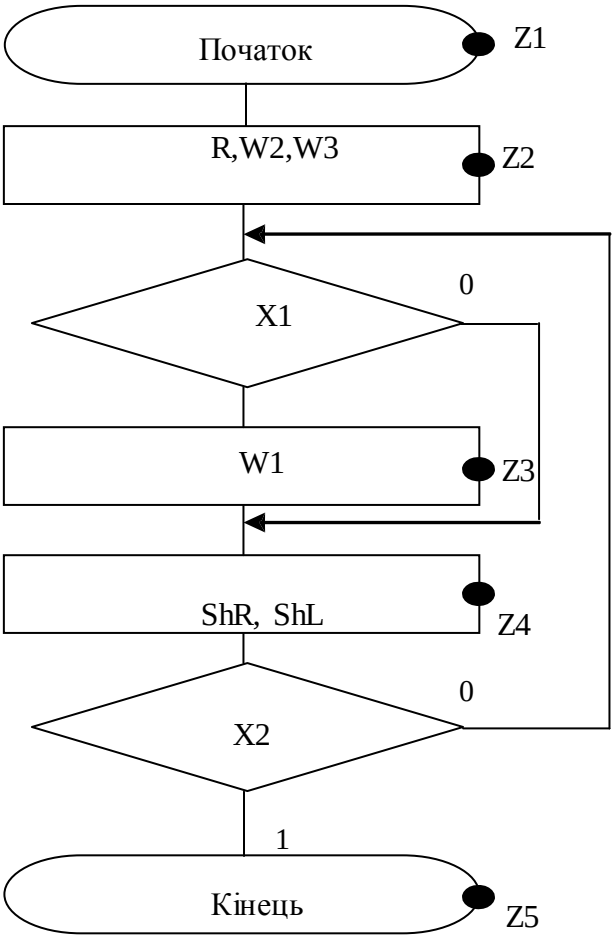

Рисунок 2.2.4-Закодований мікроалгоритм.

2.2.7 Граф управляючого автомата Мура з кодами вершин:

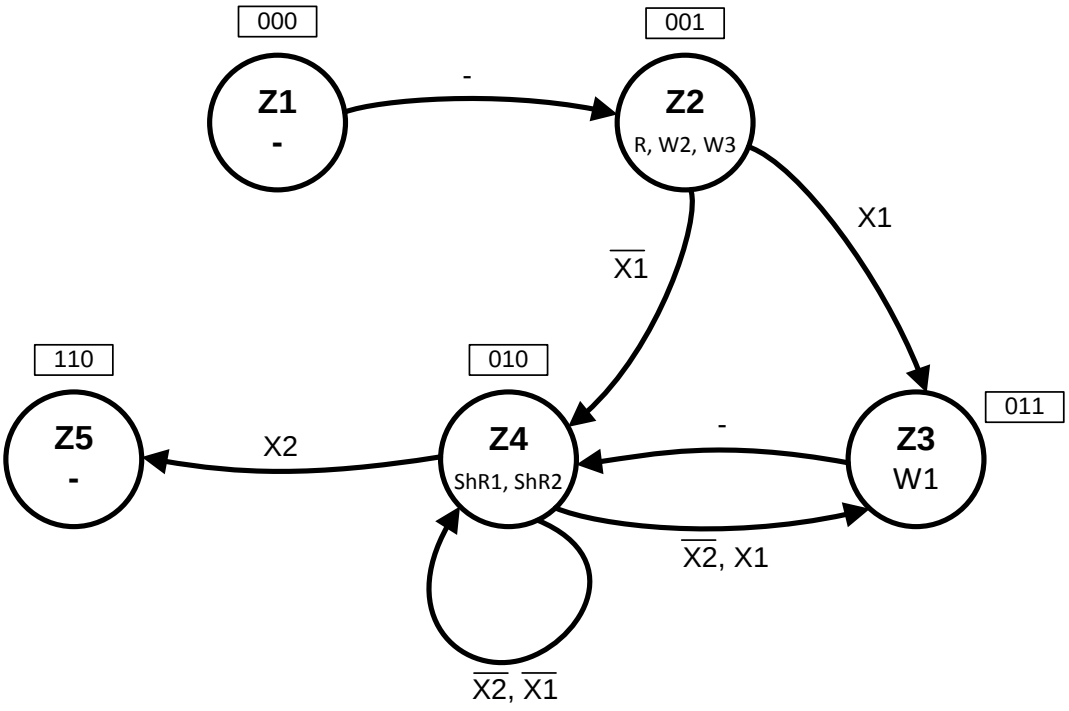

Рисунок 2.2.5 - Граф автомата Мура

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: **01100101111111010100110000111 1**

Знак мантиси: $1 \oplus 0 = 1$.

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

Порядок зменшуємо на 1:

1100101111111010100110000111 1; P_z=12;

Запишемо нормалізований результат:

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

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.

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

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

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

$Z_0 = 0$;

$Y_0 = 0$

2.3.2 Операційна схема:

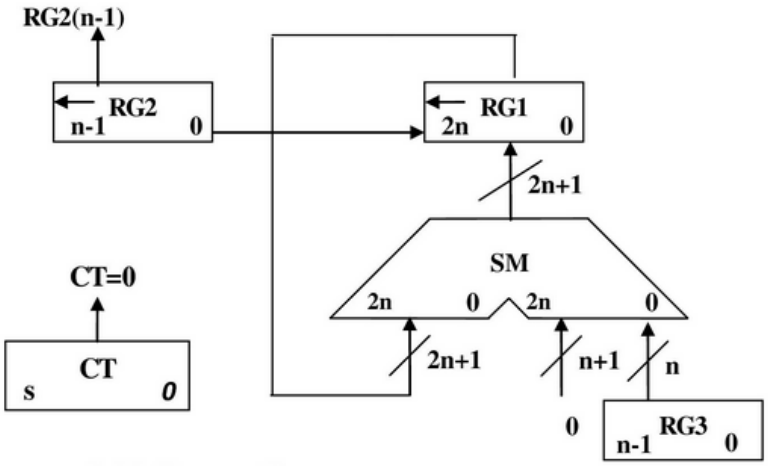

Рисунок 2.3.1 - Операційна схема

2.3.3 Змістовний мікроалгоритм:

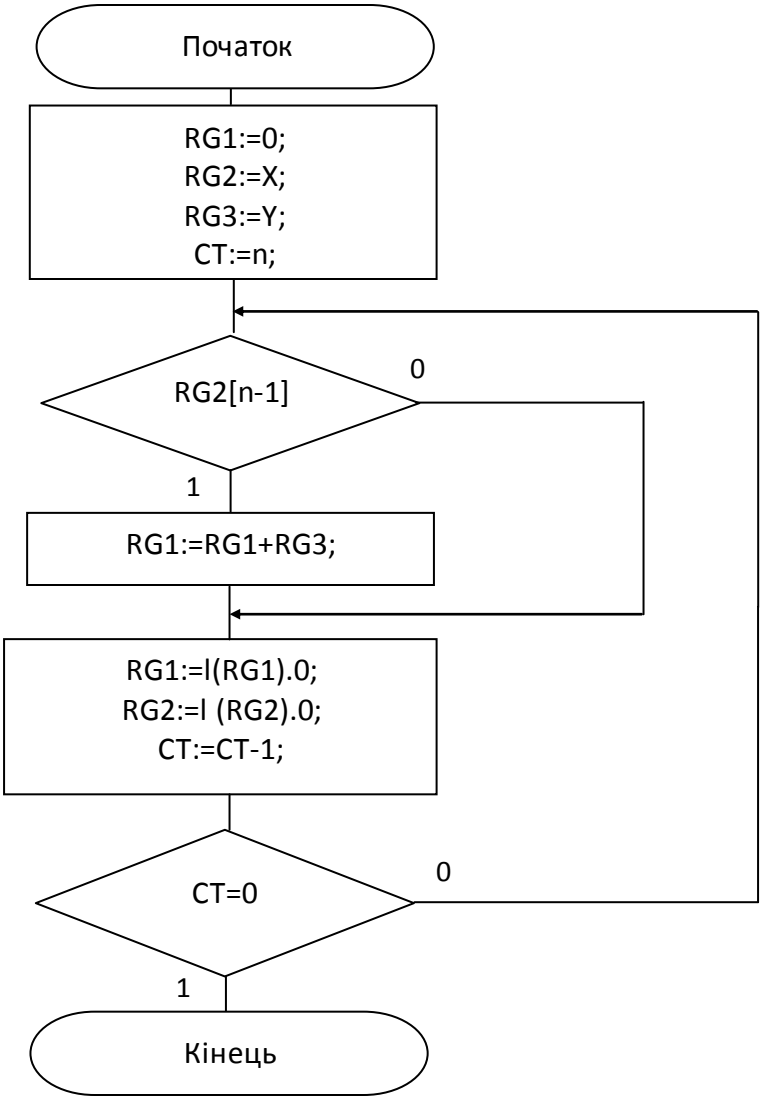

Рисунок 2.3.2 - Змістовний мікроалгоритм.

2.3.4 Таблиця станів регістрів:

Таблиця 2.3.1- Таблиця станів регістрів

№	RG1 ←	RG2 ←	RG3	CT
пс	00000000000000000000000000000000	100110011100101	101010011100011	1111
1	+00000000000000010101001110001 1 =00000000000000010101001110001 1 00000000000000101010011100011 0	001100111001010		1110
2	00000000000001010100111000110 0	011001110010100		1101
3	00000000000010101001110001100 0	110011100101000		1100
4	+00000000000000010101001110001 1 =00000000000101111110111111101 1 00000000001011111011111110110	100111001010000		1011
5	+00000000000000010101001110001 1 =00000000001100100110011011001 00000000011001001100110110010	001110010100000		1010
6	00000000110010011001101100100	011100101000000		1001
7	00000001100100110011011001000	111001010000000		1000
8	+00000000000000010101001110001 1 =00000001100101011101110101011 00000011001010111011101010110	110010100000000		0111
9	+00000000000000010101001110001 1 =00000011001011100110000111001 00000110010111001100001110010	100101000000000		0110
10	+00000000000000010101001110001 1 =00000110010111110110101010101 00001100101111101101010101010	001010000000000		0101
11	00011001011111011010101010100	010100000000000		0100
12	00110010111110110101010101000	101000000000000		0011
13	+00000000000000010101001110001 1 =00110010111111011111110001011 00110010111111011111110001011 0	010000000000000		0010
14	01100101111110111111100010110 0	100000000000000		0001
15	+00000000000000010101001110001 1 =01100101111111010100110000111 1	000000000000000		0

2.3.5 Функціональна схема:

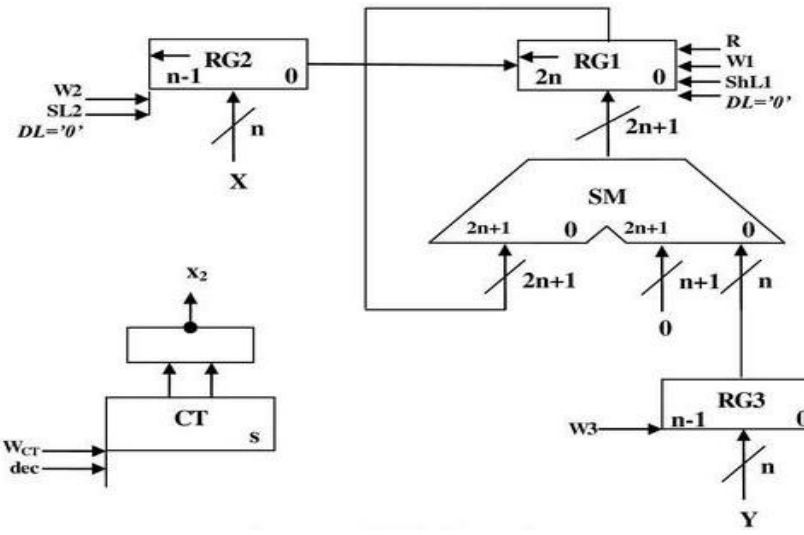

Рисунок 2.3.3 - Функціональна схема.

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		

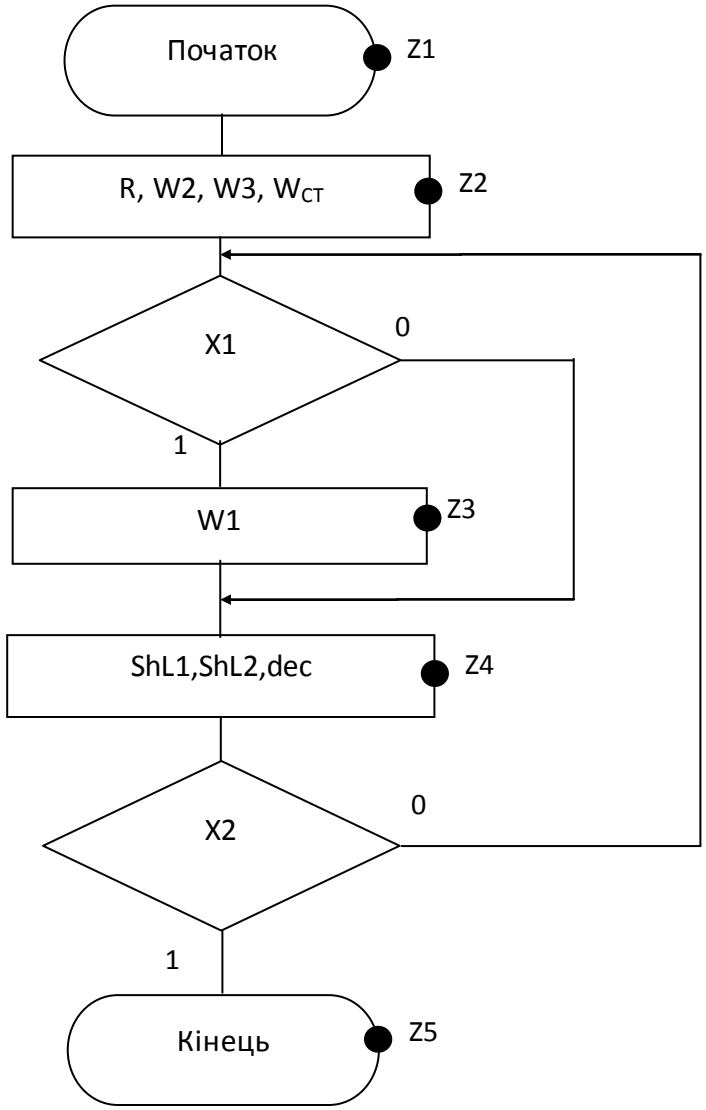

Рисунок 2.3.4-Закодований мікроалгоритм.

2.3.7 Граф управляючого автомата Мура з кодами вершин:

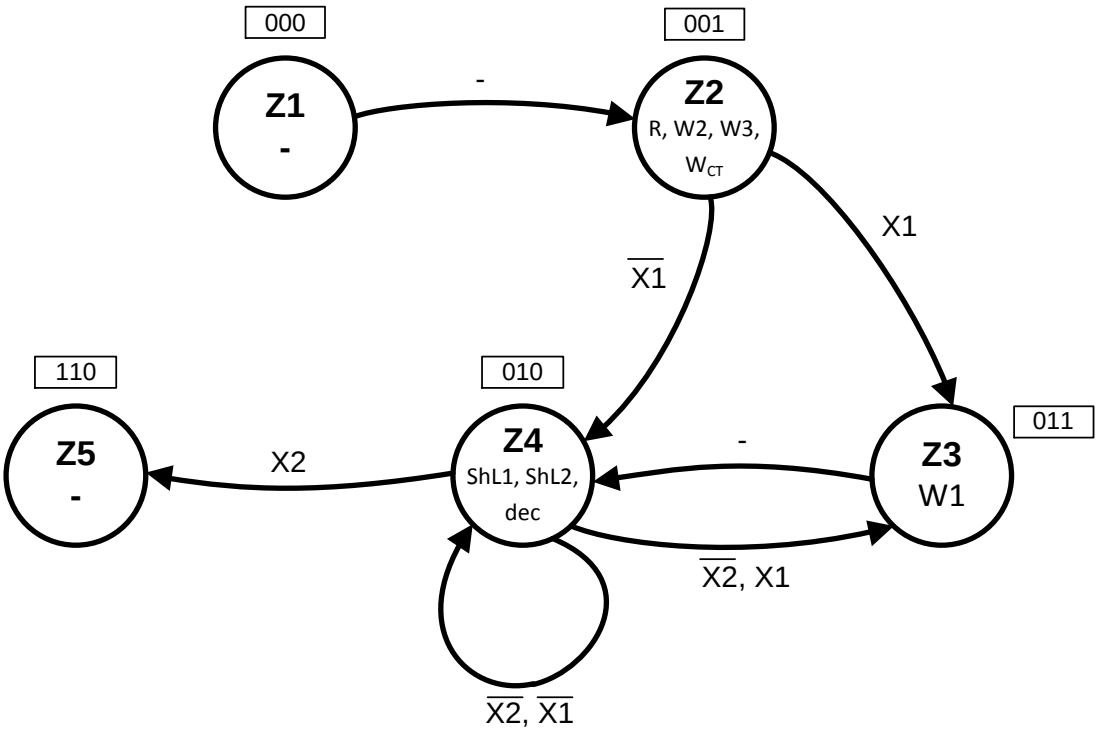

Рисунок 2.3.5 - Граф автомата Мура

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: **01100101111111010100110000111 1**

Знак мантиси: $1 \oplus 0 = 1$.

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

порядок зменшуємо на 1:

`11001011111110101001100001111`; P_z=12;

Запишемо нормалізований результат:

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

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

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

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

2.4.2 Операційна схема:

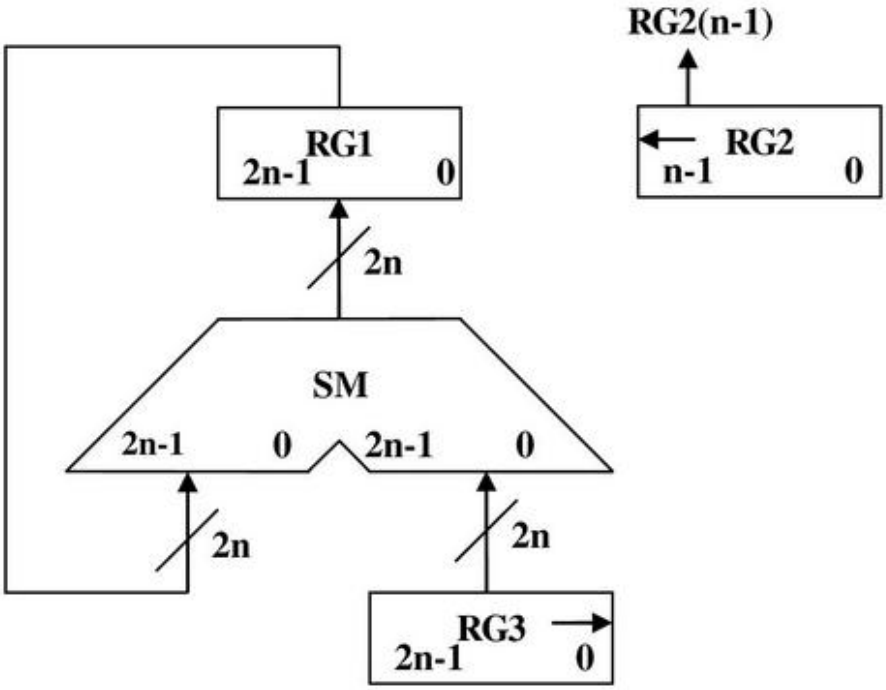

Рисунок 2.4.1- Операційна схема

2.4.3 Змістовний мікроалгоритм:

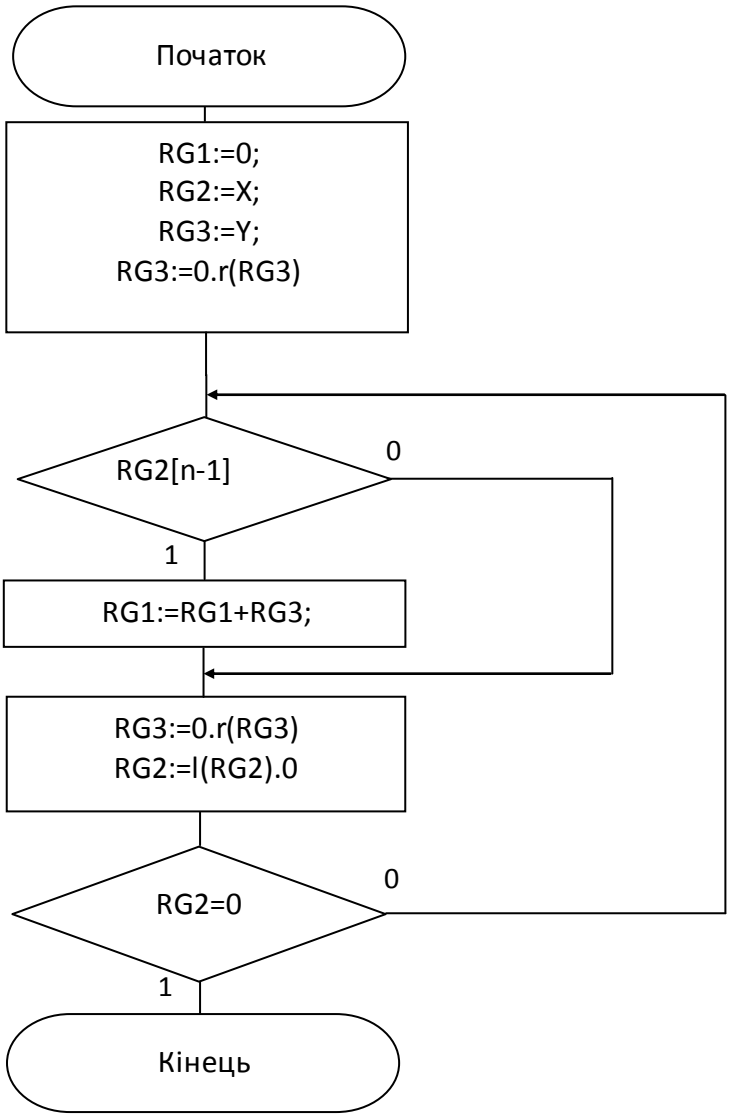

Рисунок 2.4.2 - Змістовний мікроалгоритм.

2.4.4 Таблиця станів регістрів:

Таблиця 2.4.1- Таблиця станів регістрів

№	RG1	RG3 →	RG2 ←
ПС	00000000000000000000000000000000	01010100111000110000000000000000	100110011100101
1	+01010100111000110000000000000000 =01010100111000110000000000000000	00101010011100011000000000000000	001100111001010
2	01010100111000110000000000000000	00010101001110001100000000000000	011001110010100
3	01010100111000110000000000000000	00001010100111000110000000000000	110011100101000
4	+00001010100111000110000000000000 =01011111011111110110000000000000	00000101010011100011000000000000	100111001010000
5	+00000101010011100011000000000000 =01100100110011011001000000000000	00000010101001110001100000000000	001110010100000

6	01100100110011011001000000000	00000001010100111000110000000	011100101000000
7	01100100110011011001000000000	00000000101010011100011000000	111001010000000
8	+00000000101010011100011000000 =01100101011101110101011000000	00000000010101001110001100000	110010100000000
9	+00000000010101001110001100000 =01100101110011000011100100000	00000000001010100111000110000	100101000000000
10	+00000000001010100111000110000 =01100101111101101010101000000	00000000000101010011100011000	001010000000000
11	01100101111101101010101000000	00000000000010101001110001100	010100000000000
12	01100101111101101010101000000	00000000000001010100111000110	101000000000000
13	+00000000000001010100111000110 =01100101111110111111000101100	00000000000000101010011100011	010000000000000
14	01100101111110111111000101100	00000000000000010101001110001	100000000000000
15	+00000000000000010101001110001 =01100101111111010100110000111	00000000000000001010100111000	000000000000000

2.4.5Функціональна схема:

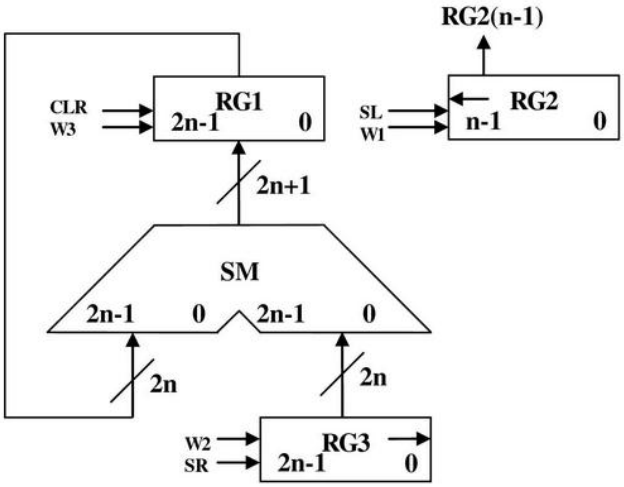

Рисунок 2.4.3 - Функціональна схема.

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		

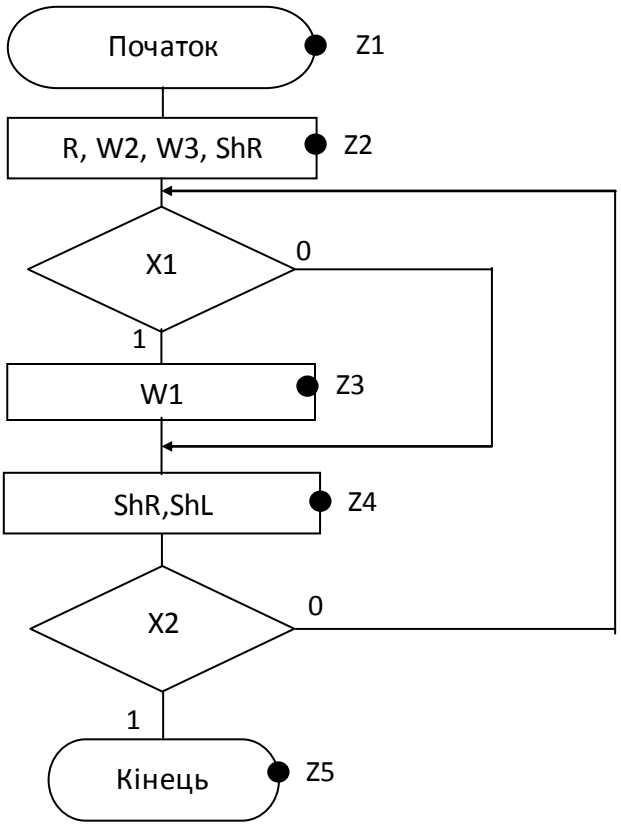

Рисунок 2.4.4-Закодований мікроалгоритм.

2.4.7 Граф управляючого автомата Мура з кодами вершин:

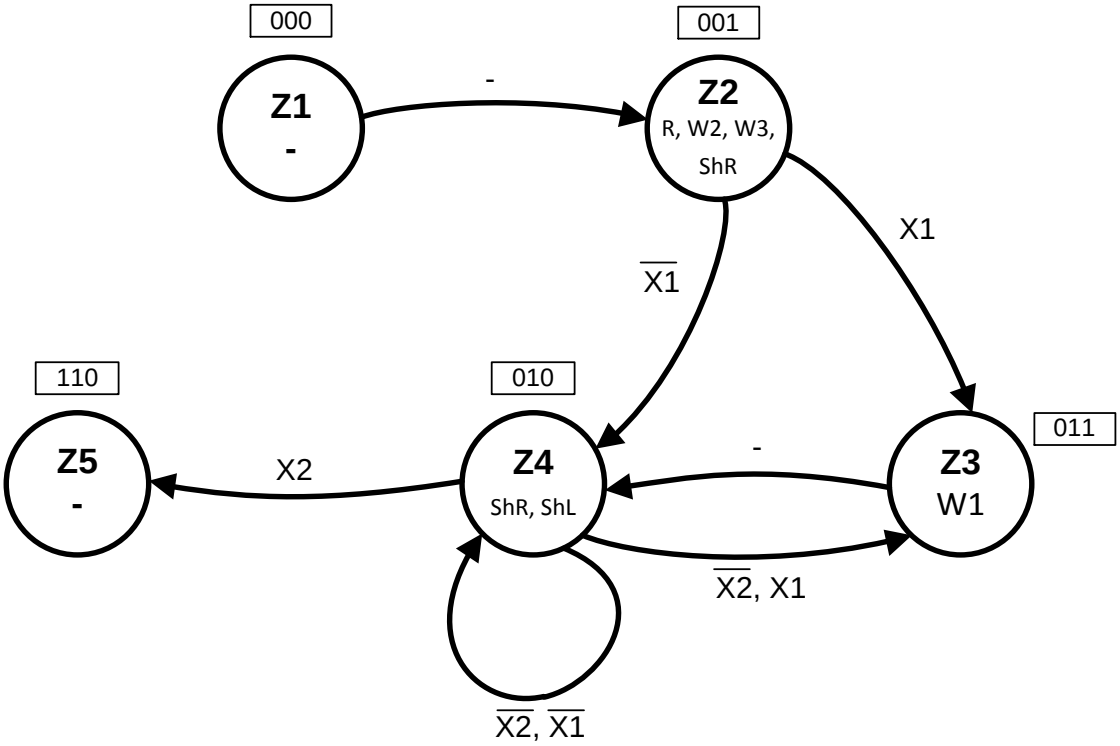

Рисунок 2.4.5 - Граф автомата Мура

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: 01100101111111010100110000111 1

Знак мантиси: $1 \oplus 0 = 1$.

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

Порядок понижаємо на 1:

11001011111110101001100001111; P_z=12;

Запишемо нормалізований результат:

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

2.5. Перший спосіб ділення.

2.5.1 Теоритичне 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:

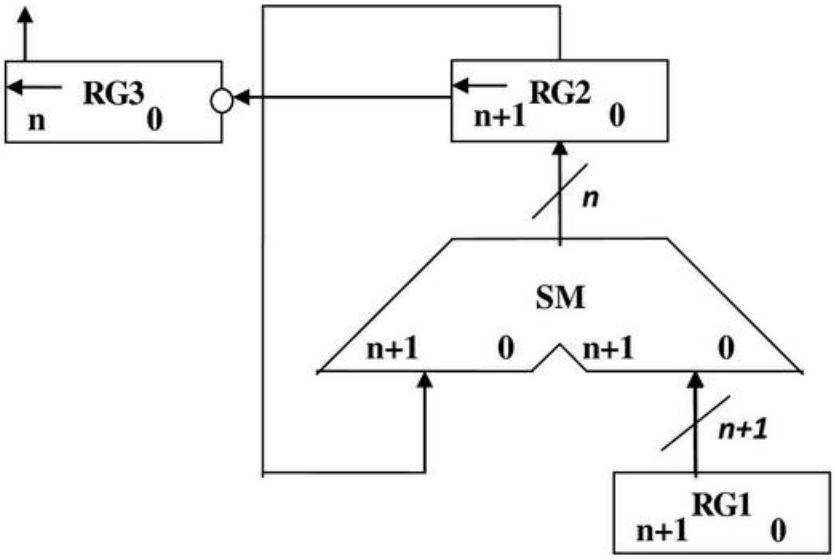

Рисунок 2.5.1- Операційна схема

2.5.3 Змістовний мікроалгоритм:

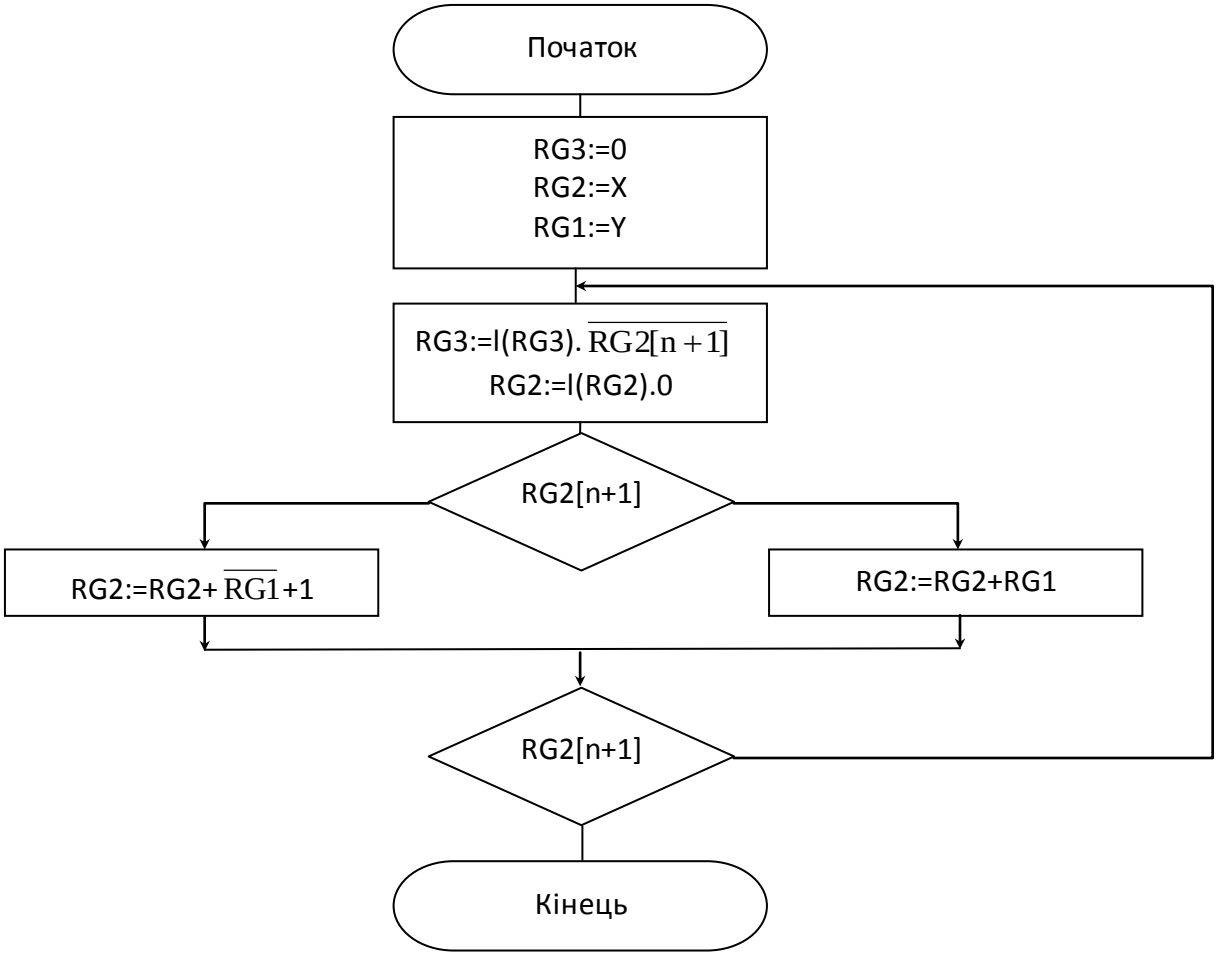

Рисунок 2.5.2-Змістовний мікроалгоритм

2.5.4 Таблиця станів регістрів:

№	RG3(Z)	RG2(X)	RG1(Y)
пс	000000000000000	00100110011100101	101010011100011
1	0000000000000001	01001100111001010 +11010101100011101 =00100010011100111	
2	0000000000000011	01000100111001110 +11010101100011101 =00011010011101011	
3	0000000000000111	00110100111010110 +11010101100011101 =00001010011110011	
4	0000000000001111	00010100111100110 +11010101100011101 =11101010100000011	
5	0000000000011110	11010101000000110 +00101010011100011 =11111111011101001	
6	0000000000111100	11111110111010010 +00101010011100011 =00101001010110101	
7	0000000001111001	01010010101101010 +11010101100011101 =00101000010000111	
8	0000000011110011	01010000100001110 +11010101100011101 =00100110000101011	
9	0000000111100111	01001100001010110 +11010101100011101 =00100001101110011	
10	0000001111001111	01000011011100110 +11010101100011101 =00011001000000011	
11	0000011110011111	00110010000000110 +11010101100011101 =00000111100100011	
12	0000111100111111	00001111001000110 +11010101100011101 =11100100101100011	
13	0001111001111110	11001001011000110 +00101010011100011 =11110011110101001	
14	0011110011111100	11100111101010010 +00101010011100011 =00010010000110101	
15	0111100111111001	00100100001101010 +11010101100011101 =11111001110000111	
16	**1111001111110010**	11110011100001110 +00101010011100011 =00011101111110001	

2.5.5 Функціональна схема:

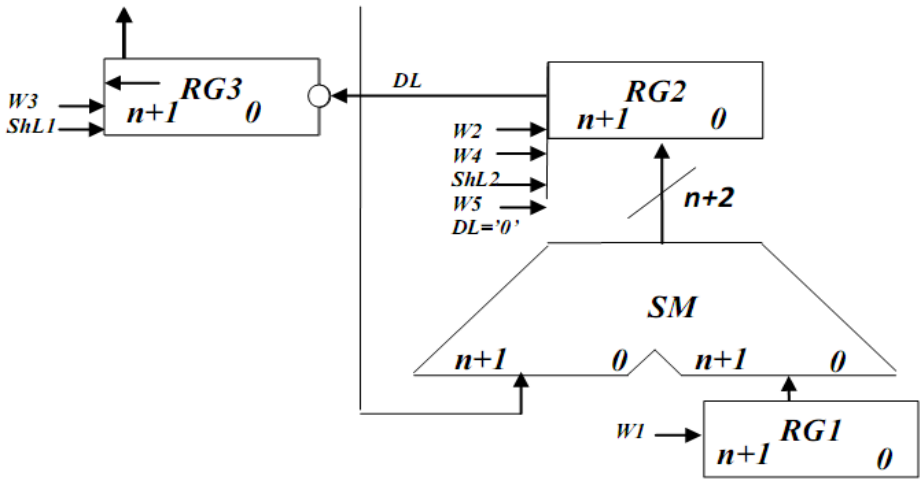

Рисунок 2.5.3 – Функціональна схема

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; RG3:=l(RG3).$\overline{RG2[n+1]}$ RG2:=l(RG2).0 RG2:=RG2+$\overline{RG1}$+1 RG2:=RG2+RG1	W3 W2 W1 ShL1 ShL2 W4 W5	RG2[n-1] RG2=0	X1 X2

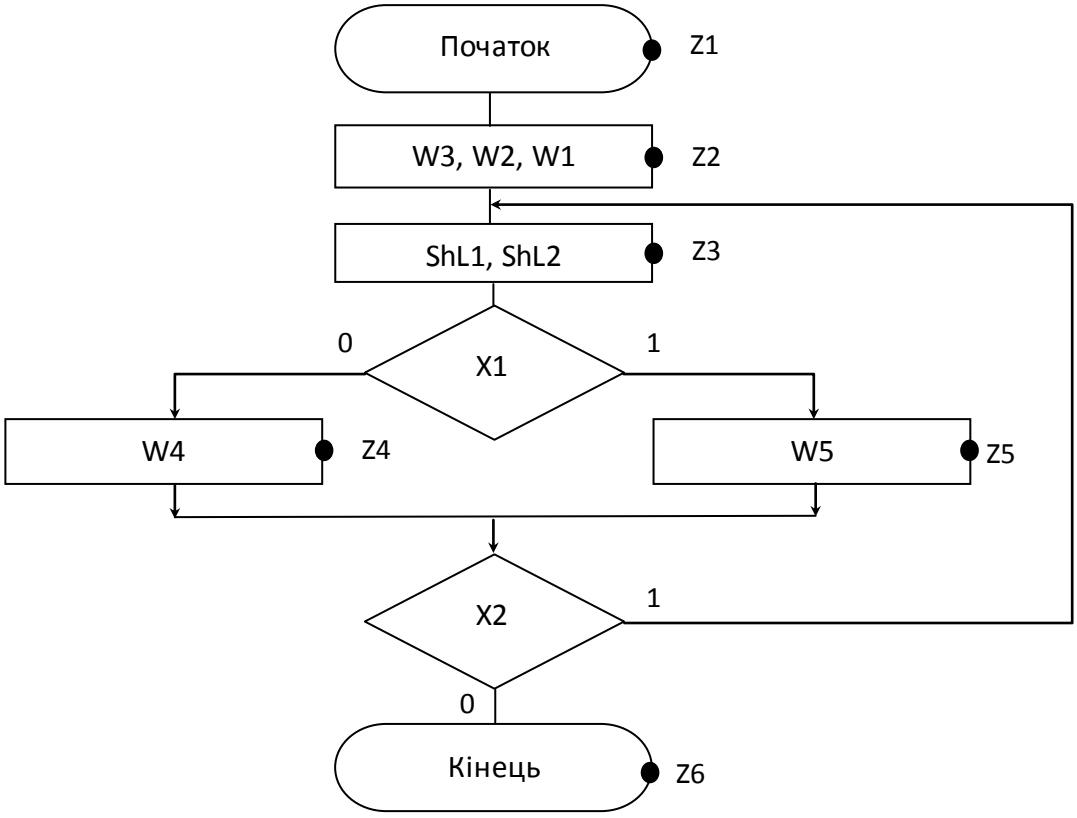

Рисунок 2.5.4-Закодований мікроалгоритм.

2.5.7 Граф управляючого автомата Мура з кодами вершин:

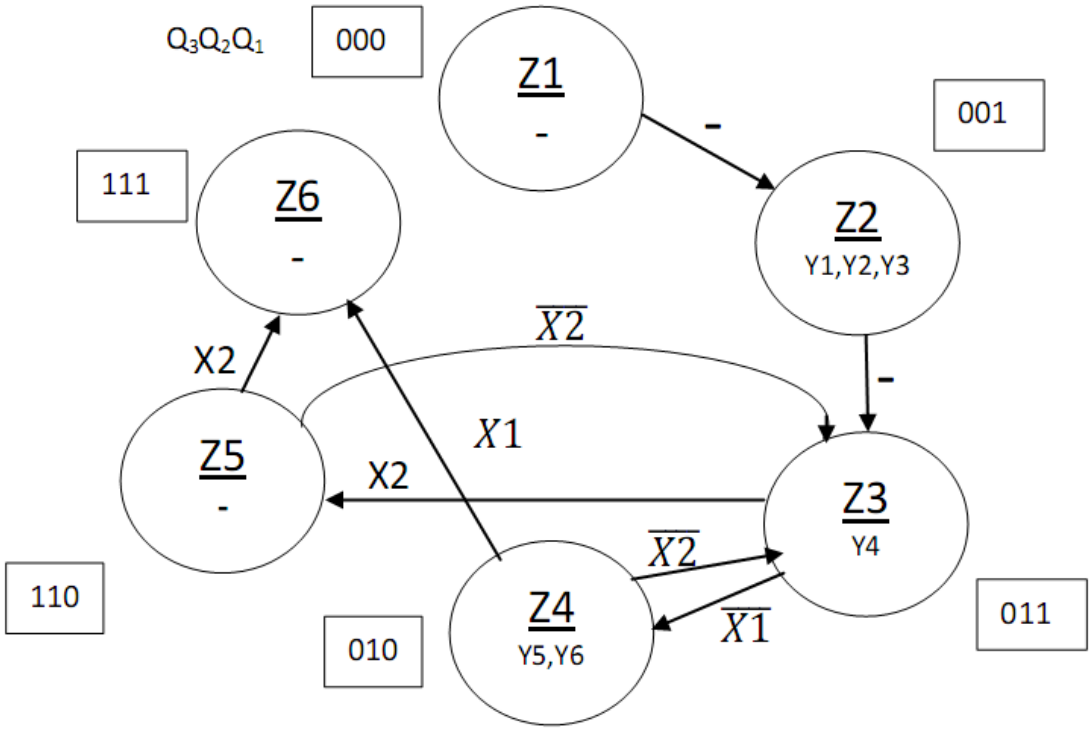

Рисунок 2.5.5 - Граф управляючого автомата.

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: **1111001111110010**

Знак мантиси: $1 \oplus 0 = 1$.

Нормалізація мантиси не потрібна.

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

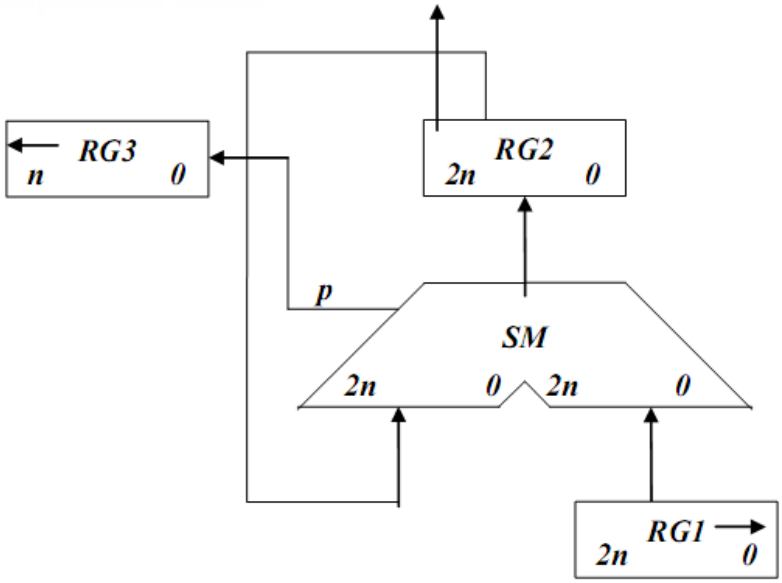

Рисунок 2.6.1-Операційна схема

2.6.3 Змістовний мікроалгоритм

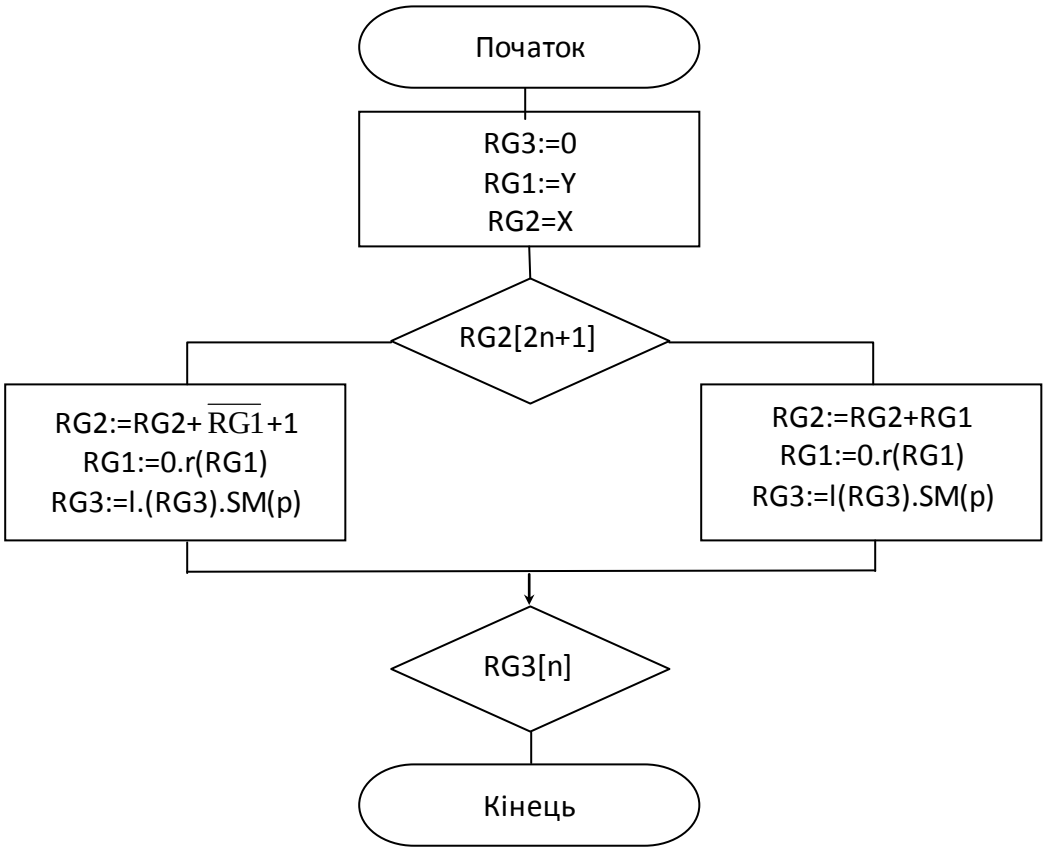

Рисунок 2.6.2-Змістовний мікроалгоритм

2.6.4 Таблиця станів регістрів

Таблиця 2.6.1- Таблиця станів регістрів

№	RG3 (Z)	RG2 (X)	RG1 (Y)
пс	0000000000000001	01001100111001010000000000000000	00101010011100011000000000000000
1	0000000000000011	01001100111001010000000000000000 +11010101100011101000000000000000 =00100010011100111000000000000000	00010101001110001100000000000000
2	0000000000000111	00100010011100111000000000000000 +11101010110001110100000000000000 =00001101001110101100000000000000	00001010100111000110000000000000
3	0000000000001111	00001101001110101100000000000000 +11110101011000111010000000000000 =00000010100111100110000000000000	00000101010011100011000000000000
4	0000000000011110	00000010100111100110000000000000 +11111010101100011101000000000000 =11111101010100000011000000000000	00000010101001110001100000000000
5	0000000000111100	11111101010100000011000000000000 +00000010101001110001100000000000 =11111111111101110100100000000000	00000001010100111000110000000000
6	0000000001111001	11111111111101110100100000000000 +00000001010011100011000000000000 =00000001010010101101010000000000	00000000101010011100011000000000
7	0000000011110011	00000001010010101101010000000000 +11111111010101100011101000000000 =00000000101000010000111000000000	00000000010101001110001100000000
8	0000000111100111	00000000101000010000111000000000 +11111111101010110001110100000000 =00000000010011000010101100000000	00000000001010100111000110000000
9	0000001111001111	00000000010011000010101100000000 +11111111110101011000111010000000 =00000000001000011011100110000000	00000000000101010011100011000000
10	0000011110011111	00000000001000011011100110000000 +11111111111010101100011101000000 =00000000000011001000000110000000	00000000000010101001110001100000
11	0000111100111111	00000000000011001000000110000000 +11111111111101010110001110100000 =00000000000000011110010001100000	00000000000001010100111000110000
12	0001111001111110	00000000000000011110010001100000 +11111111111110101011000111010000 =11111111111111001001011000110000	00000000000000101010011100011000
13	0011110011111100	11111111111111001001011000110000 +00000000000000101010011100011000 =11111111111111111001111010100010	00000000000000010101001110001100
14	0111100111111001	11111111111111111001111010100010 +00000000000000001010100111000110 =00000000000000000100100001101010	00000000000000000101010011100011
15	**1111001111110010**	00000000000000000100100001101010 +11111111111111111010110001111000 =11111111111111111111001110001000	00000000000000000010101001110001

2.6.5 Функціональна схема з відображенням управляючих сигналів

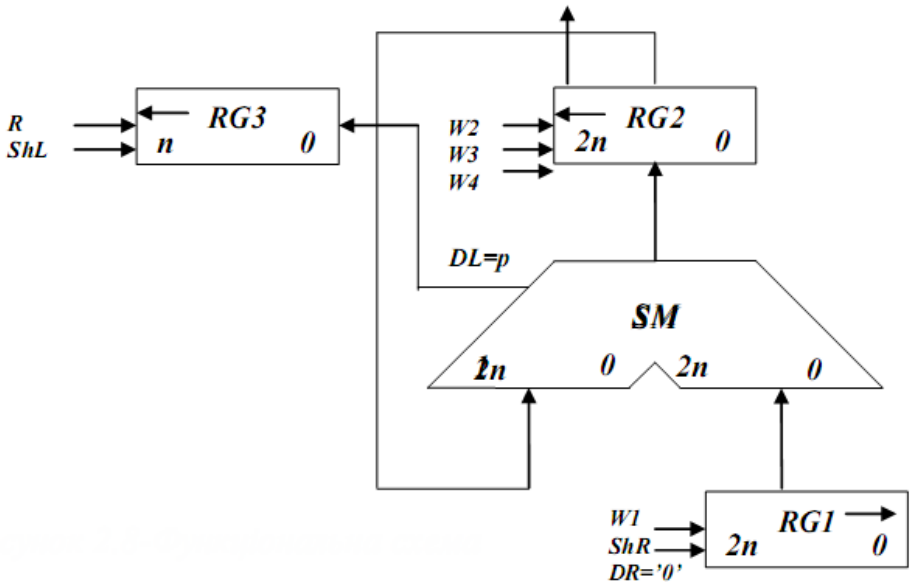

Рисунок 2.6.3-Функціональна схема

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

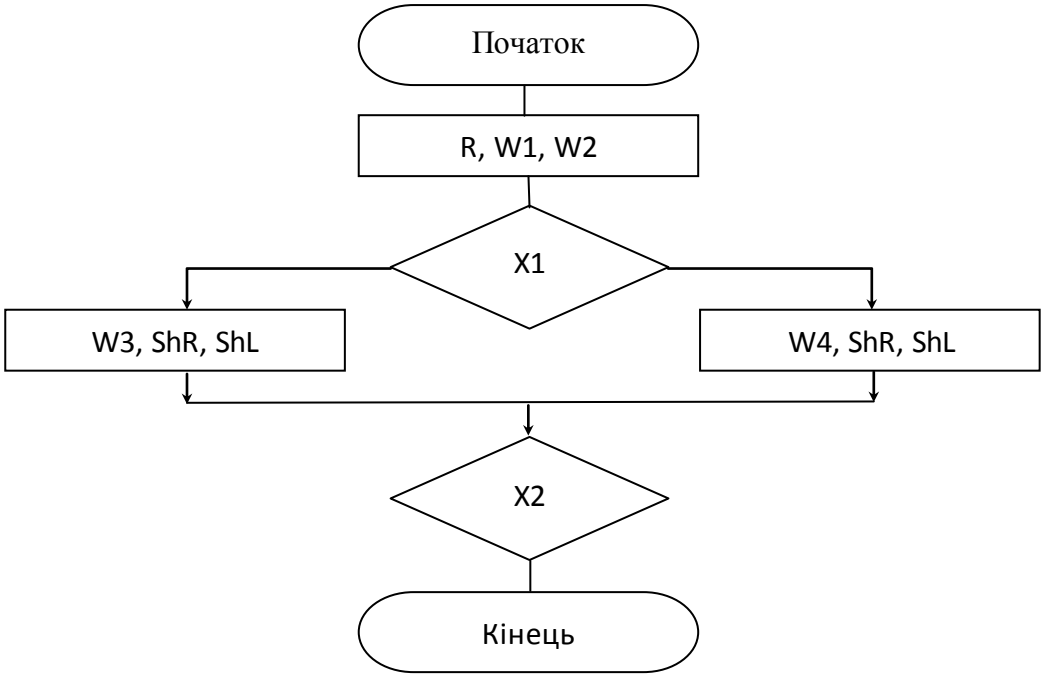

Рисунок 2.6.4- Закодований мікроалгоритм

2.6.7 Граф управляючого автомата Мура з кодами вершин

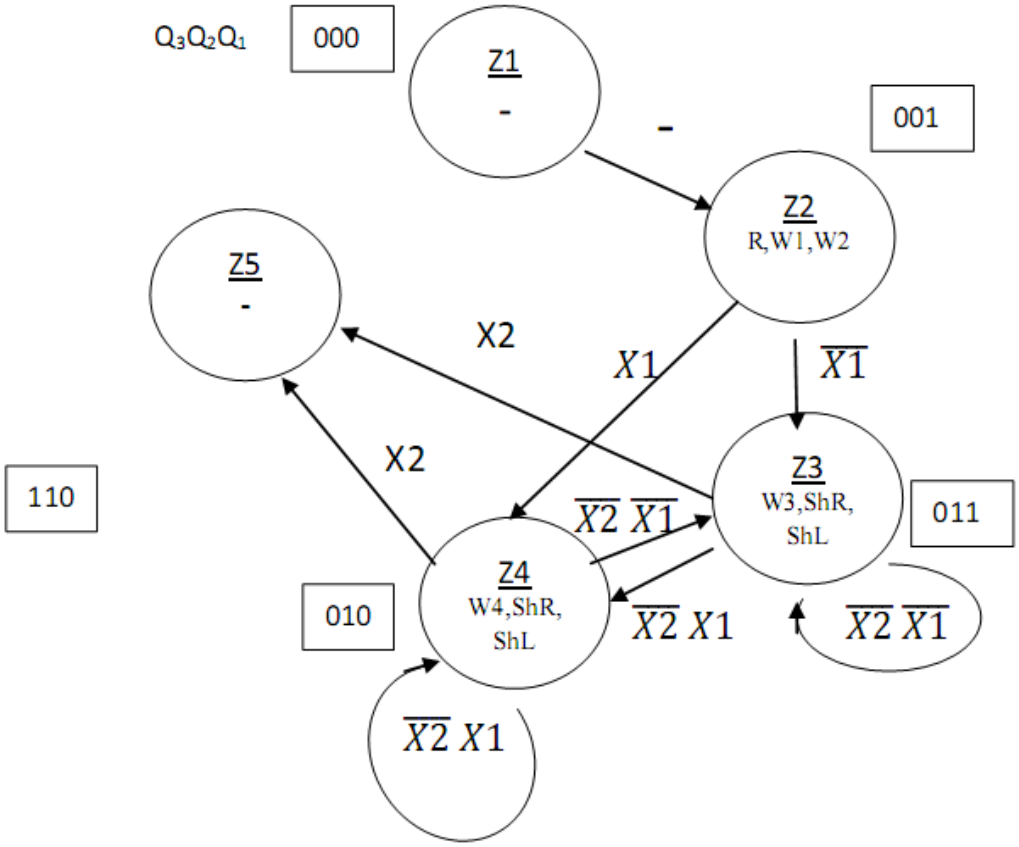

Рисунок 2.6.5- Граф автомата Мура

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;

2.6.9 Нормалізація результату:

Отримали результат: **1111001111110010**

Знак мантиси: $1 \oplus 0 = 1$.

Нормалізація мантиси не потрібна.

0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	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, 101010011100011	11	Початковий стан
0, 010101001110001	10	M_y= 0.r(M_y); Δ:=Δ-1
0, 001010100111000	01	M_y= 0.r(M_y); Δ:=Δ-1
0,000101010011100	00	M_y= 0.r(M_y); Δ:=Δ-1

3. Додавання мантис у модифікованому ДК.

$X_{мдк}$ = 11. 011001100011010

$Y_{мдк}$ = 00. 000101010111000

Таблиця 2.7.2-Додавання мантис(для додавання)

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

$Z_{пк}$ = 1. 100001001001001

4. Нормалізація результату (В ПК).

Для даного результату додавання нормалізація не потрібна.

2.7.2 Операційна схема

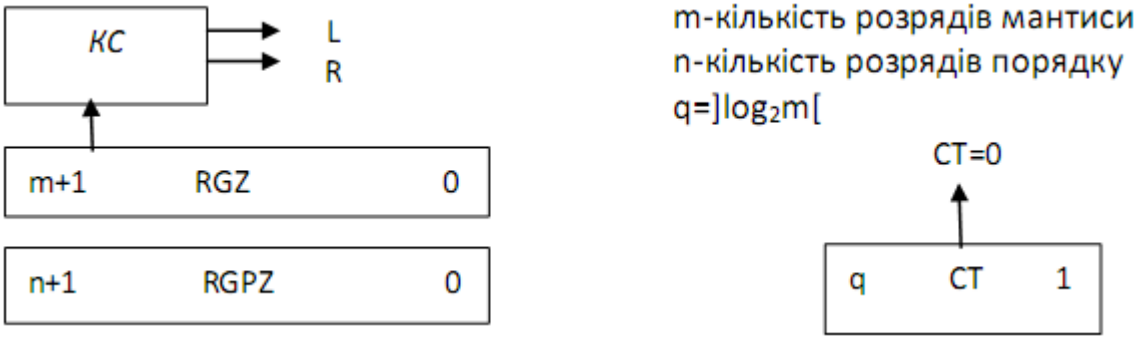

Рисунок 2.7.1-Операційна схема

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

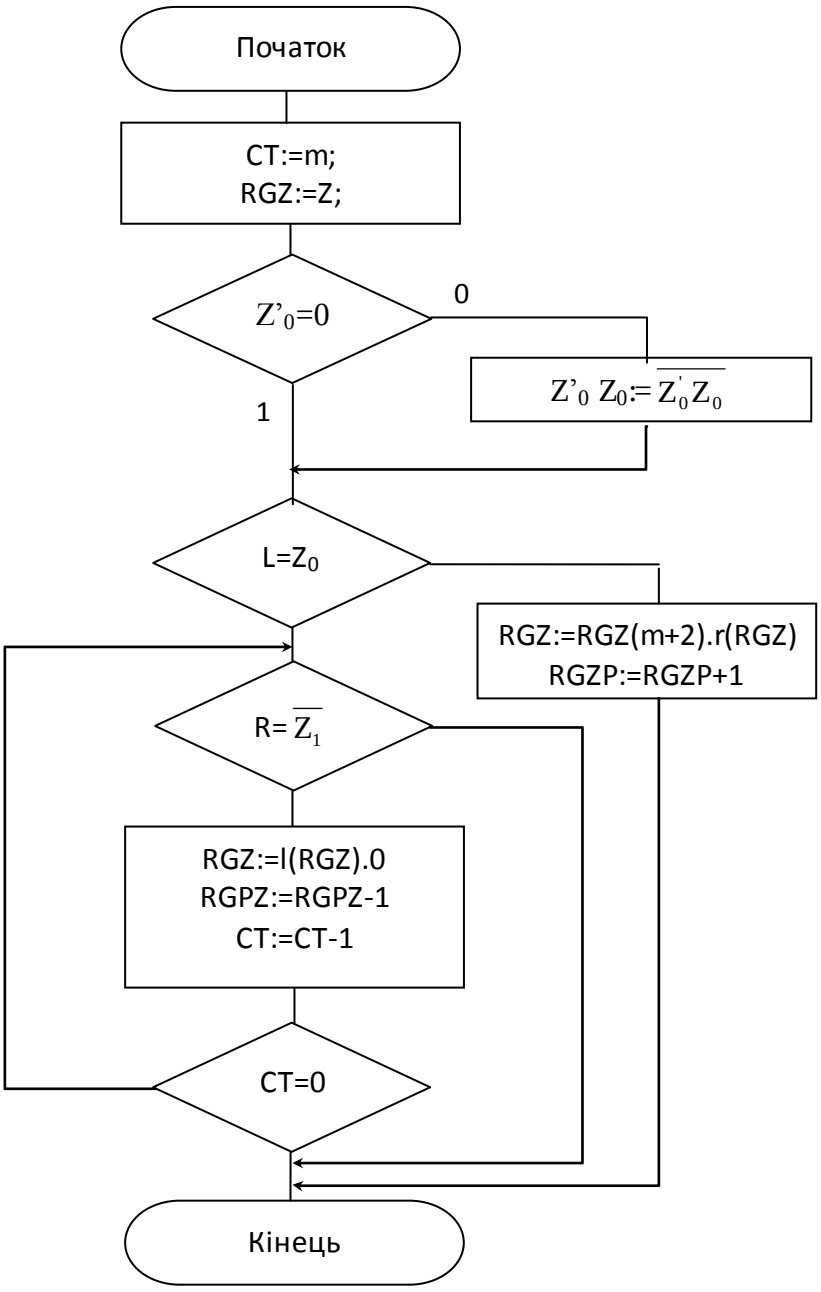

Рисунок 2.7.2-Змістовний мікроалгоритм

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	00.111110011001001	0	1	100	
1	000111	00.111100110010010 00.111001100100100	0	0	011	$Z'_0\ Z_0 := \overline{Z'_0}\ \overline{Z_0}$ RGZ:=l(RGZ).0 RGPZ:=RGPZ-1 CT:=CT-1

2.7.5 Функціональна схема з відображенням керуючих сигналів

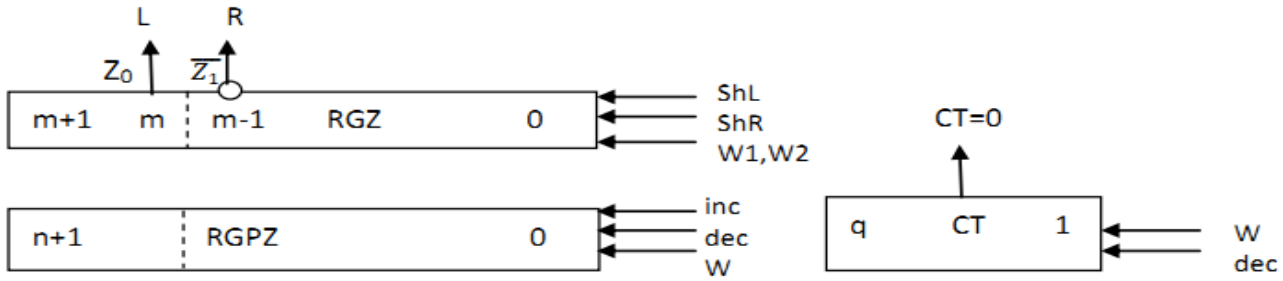

Рисунок 2.7.3 – Функціональна схема

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

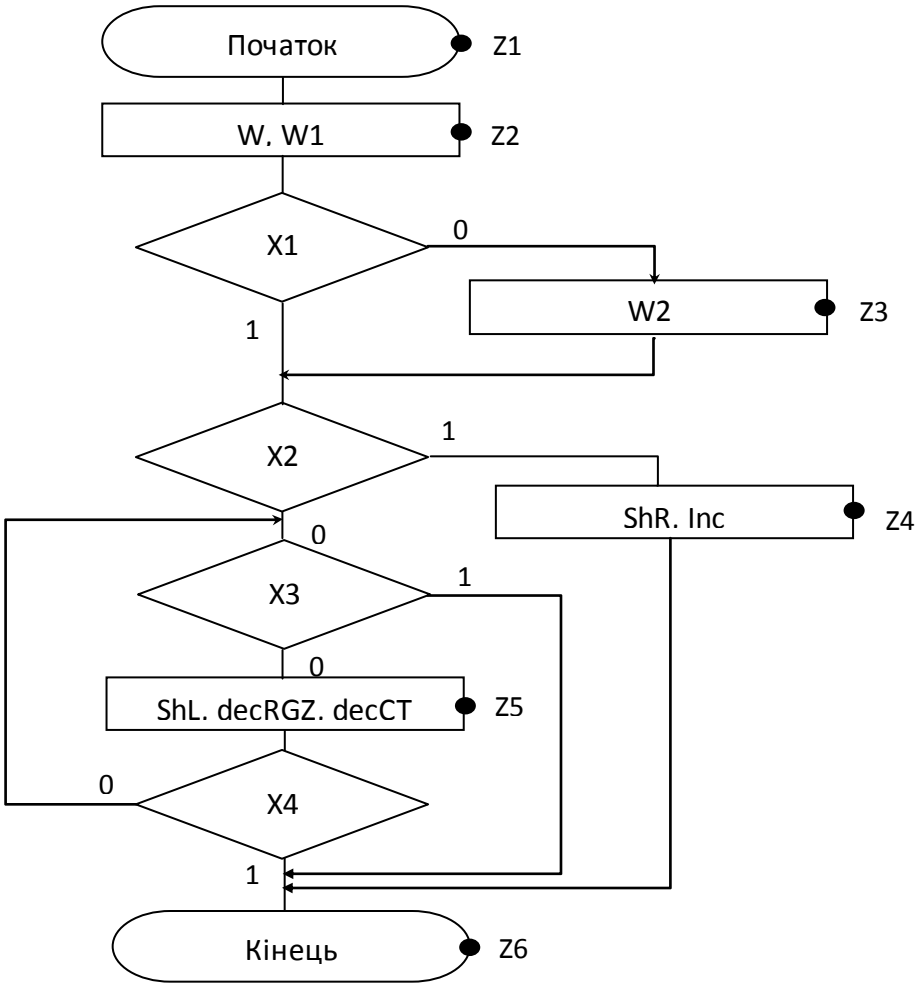

Рисунок 2.7.4 – Закодований мікроалгоритм

2.7.7 Граф управляючого автомата Мура з кодами вершин

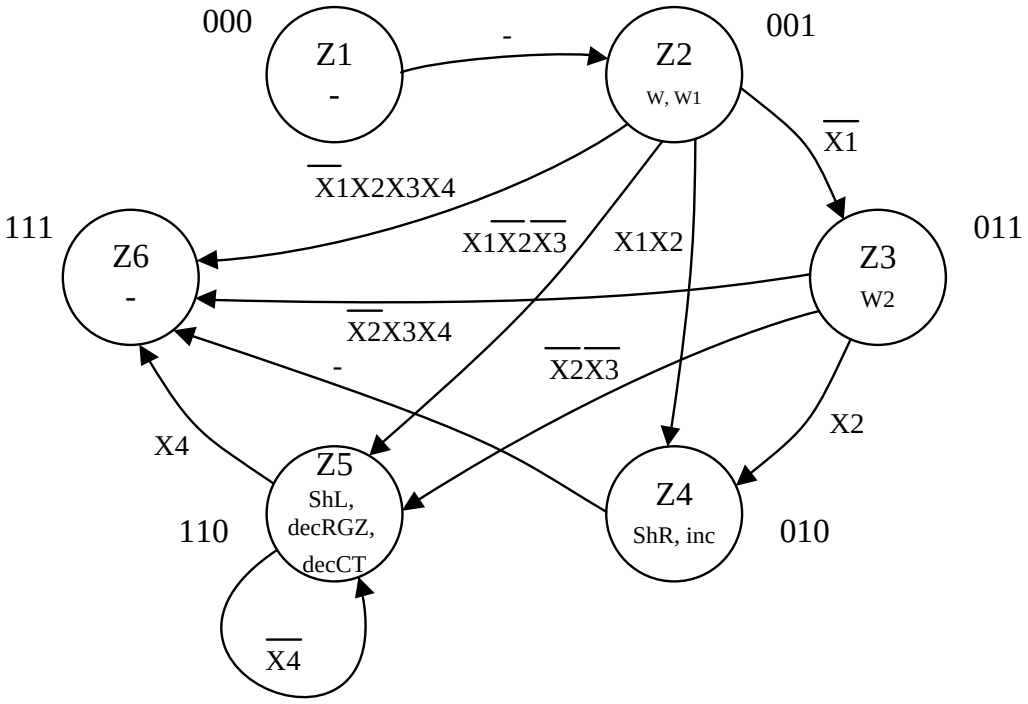

Рисунок 2.7.5 – Граф автомата Мура

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.

$Z_{пк} = 1.100001001001001$

$P_z = 8_{10} =1000_2$ $M_z = 100001001001001_2$

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

2.8.Операціядобування кореня

2.8.1Теоритичнеобгрунтуванняоперації 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

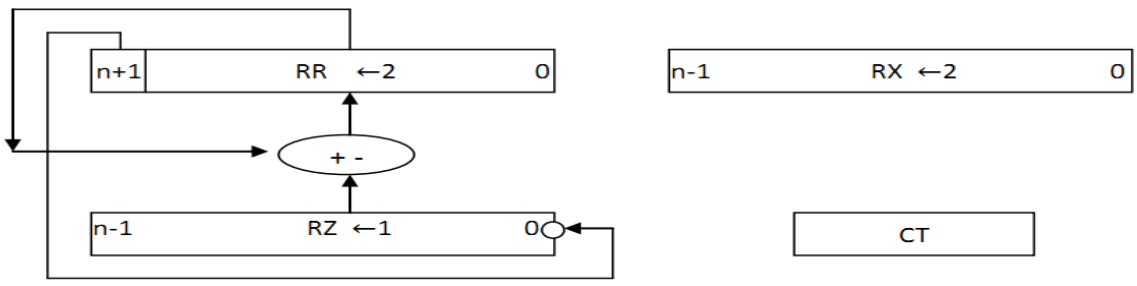

Рисунок 2.8.1 –Операційна схема

2.8.3 Змістовний мікроалгоритм

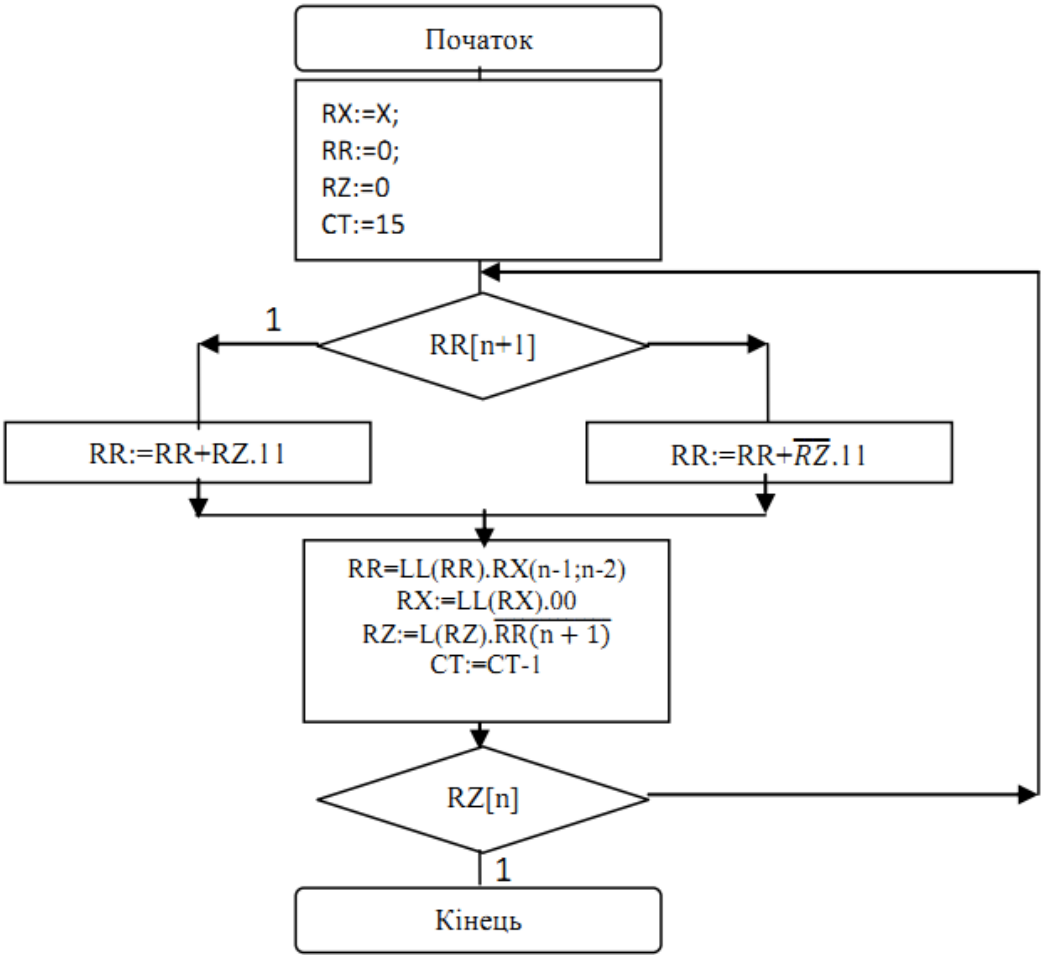

Рисунок 2.8.2 – Змістовний мікроалгоритм

2.8.4 Таблиця станів регістрів

Таблиця 2.8.1 – Таблиця станів регістрів

№	RZ	RR	RX	CT
пс		0000000000000000		
пз	000000000000000	0000000000000010	100111011101101	1111
1	000000000000001	+1111111111111111 =0000000000000001 0000000000000101	010011001110010	1110
2	000000000000011	+1111111111111011 =0000000000000000 0000000000000010	001001100111001	1101
3	000000000000110	+1111111111110011 =1111111111110101 1111111111010101	000100110011100	1100
4	000000000001100	+0000000000011011 =1111111111110000 1111111111000011	000010011001110	1011
5	000000000011000	+0000000000110011 =1111111111110110 1111111111011000	000001001100111	1010
6	000000000110001	+0000000001100011	000000100110011	1001

		=0000000000111011 0000000011101110		
7	000000001100011	+1111111100111011 =0000000000101001 0000000010100110	000000010011001	1000
8	000000011000110	+1111111001110011 =1111111100011001 1111111000110100	000000001001100	0111
9	000000110001100	+0000000110001101 =1111111101111111 1111111011111100	000000000100110	0110
10	000001100011001	+0000001100011001 =0000001000010111 0000100001011100	000000000010011	0101
11	000011000110011	+1111100111001101 =0000001000101011 0000100010101100	000000000001001	0100
12	000110001100110	+1111001110011001 =1111110001000111 1111000100011100	000000000000100	0011
13	001100011001101	+0001100011001101 =0000100111010111 0010011110101100	000000000000010	0010
14	011000110011010	+1100111001100101 =1111011000010011 1101100001001100	000000000000001	0001
15	**110001100110101**	+0110001100110101 =0011101110000011 1110111000011100	000000000000000	0000

2.8.5Функціональна схема операції обчислення квадратного кореня

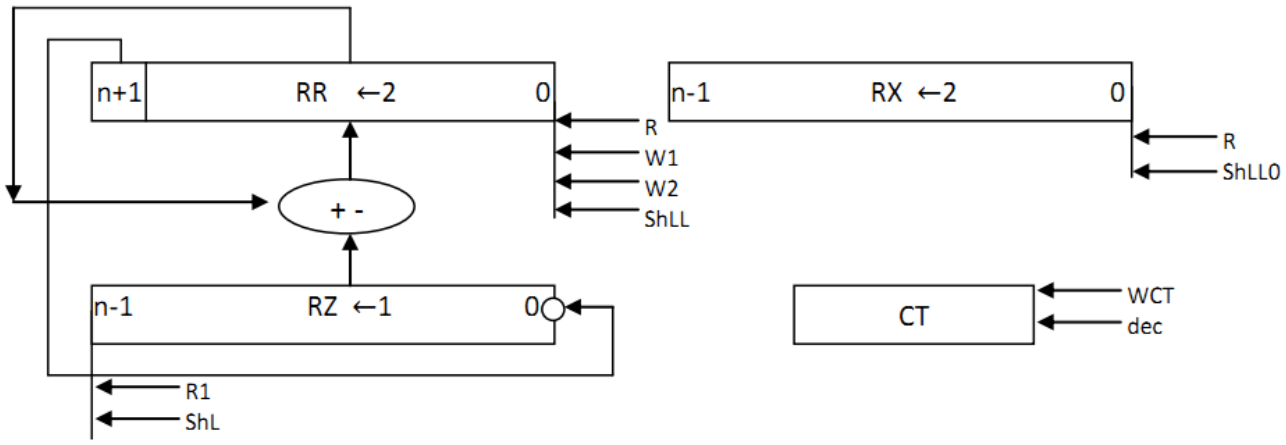

Рисунок 2.8.3 – Функціональна схема

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

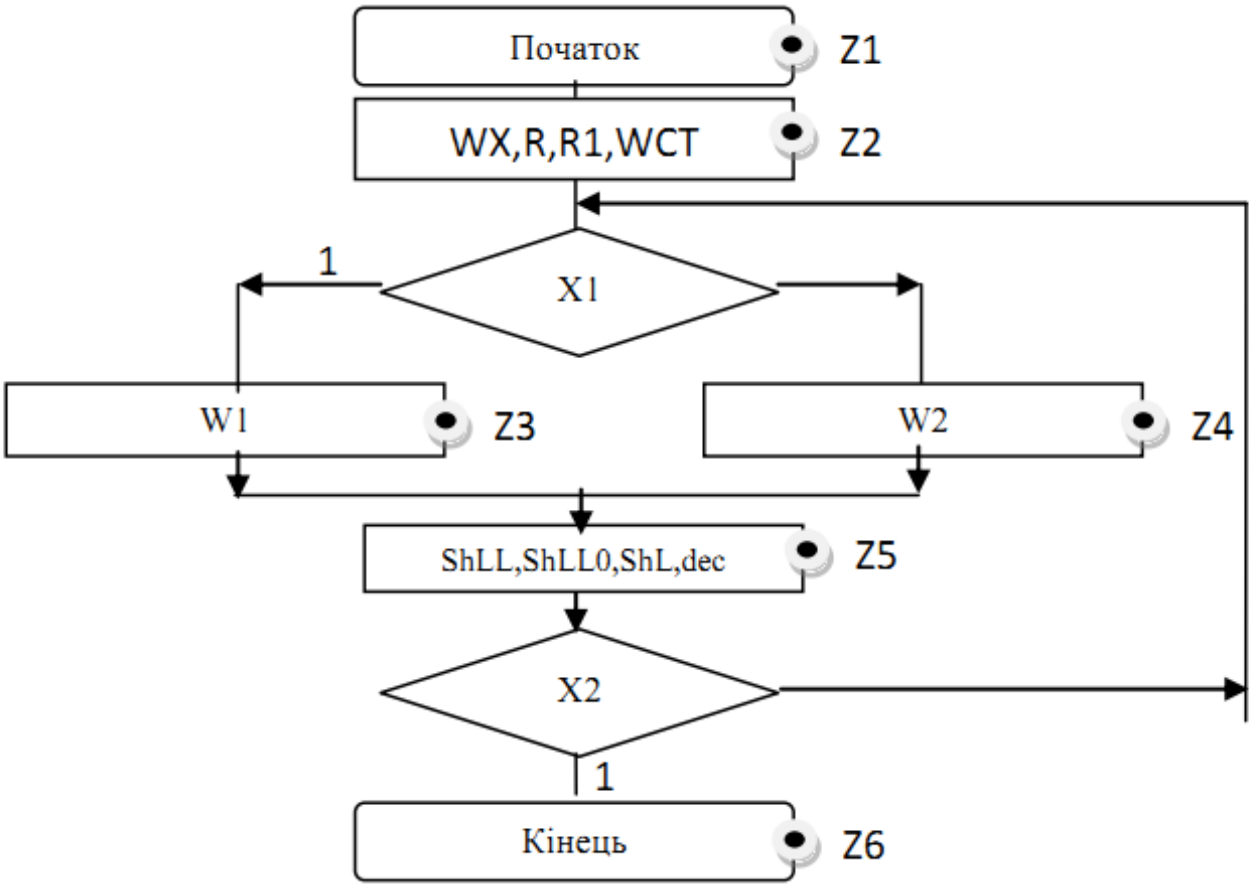

Рисунок 2.8.4 – Закодований мікроалгоритм

2.8.7Граф управляючого автомата Мура з кодами вершин

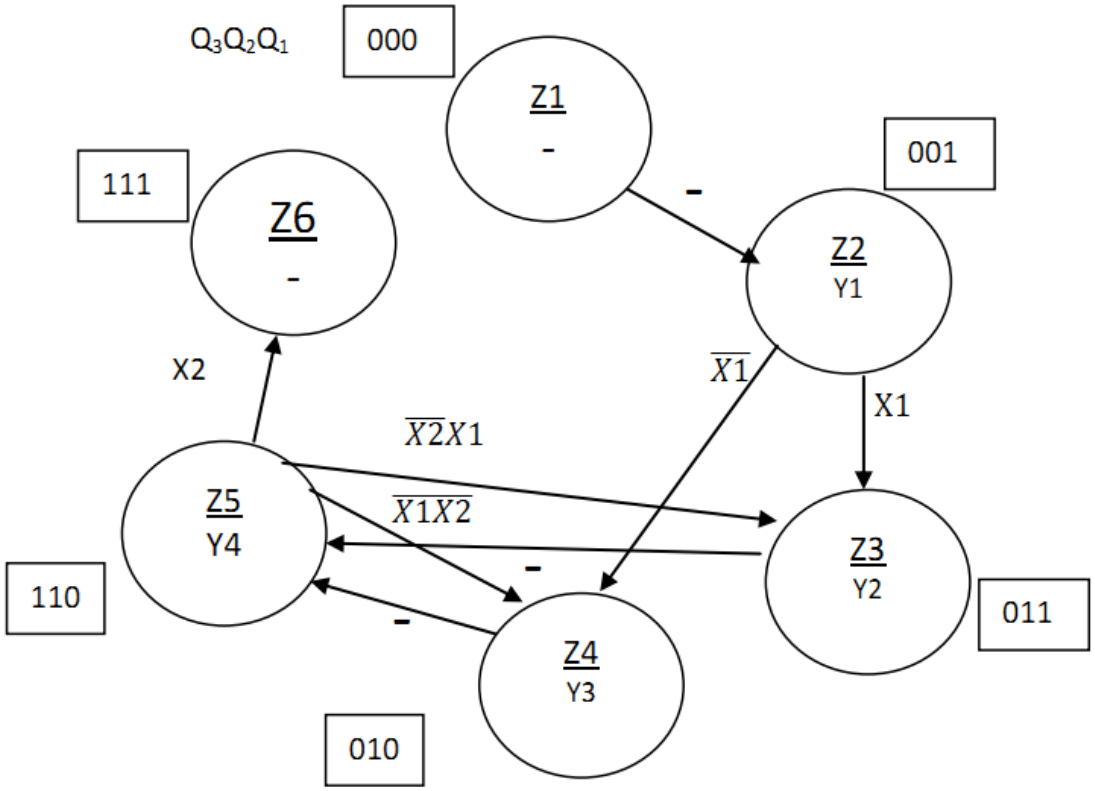

Рисунок 2.8.5 – Граф управляючого автомата Мура

2.8.8 Обробка порядків

$$P_z = P_x/2;$$

В моєму випадку P_z=4;

2.8.9Запис результату

Отримали результат Z = **110001100110101**;

Результат нормалізований, готовий до запису у мантису:

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

Завдання 3

$x_3\ x_2\ x_1 + 1 = 010_2 = 2_{10}$.

Синтез управляючого автомату Мура на D-тригерах для операції множення другим способом

3.1 Таблиця кодування сигналів

Таблиця 3.1 – Таблиця кодування сигналів

R,W2,W3	Y1
W1	Y2
ShR, ShL	Y3

3.2 Мікроалгоритм в термінах управляючого автомата

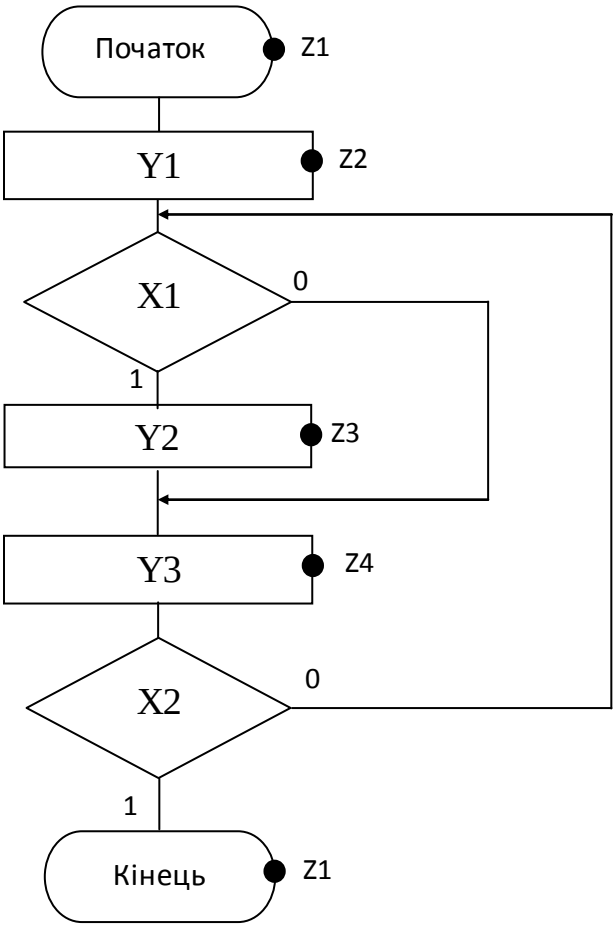

Рисунок 3.1 – Закодований мікроалгоритм

3.3 Граф автомата

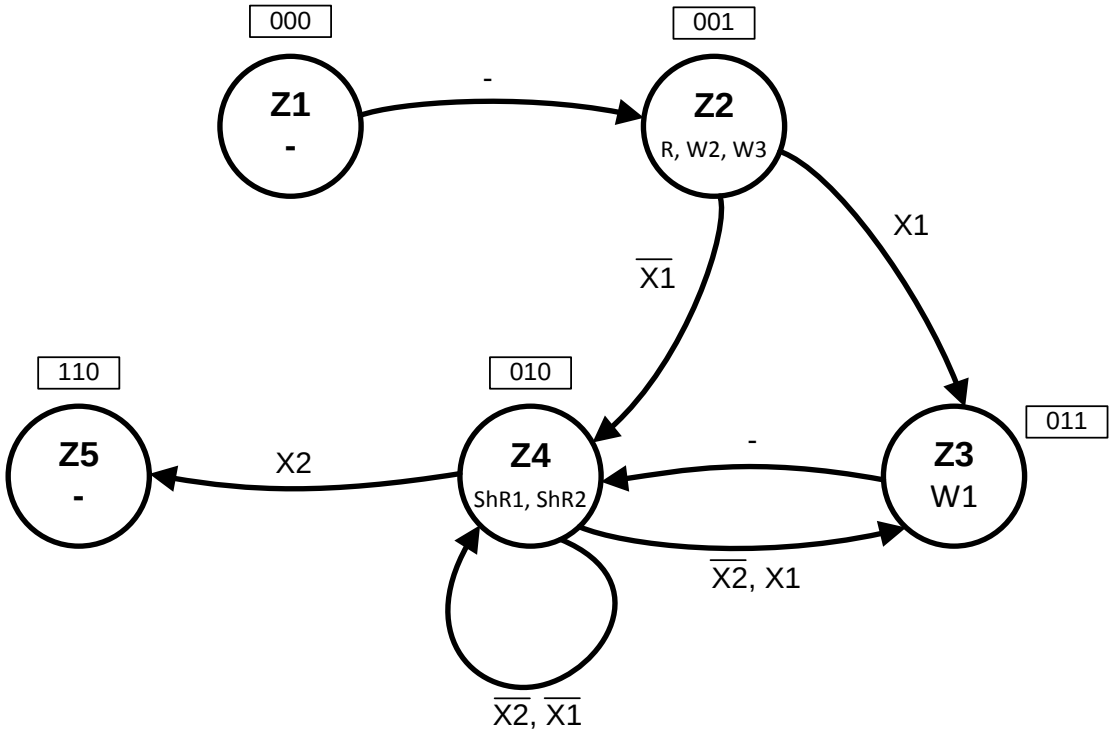

Рисунок 3.2 – Граф циклічного автомата

3.4 Таблиця переходів циклічного автомата на D-тригерах

Таблиця 3.2 – Таблиця переходів

Пер.	Ст. ст.	Нов. стан	Вх. сигн.	Вих. сигн.	Функції тригерів		
	$Q_3Q_2Q_1$	$Q_3Q_2Q_1$	X_2X_1	$Y_1Y_2Y_3$	D_3	D_2	D_1
$Z_1 \rightarrow Z_2$	000	001	- -	0 0 0	0	0	1
$Z_2 \rightarrow Z_3$	001	011	- 1	1 0 0	0	1	1
$Z_2 \rightarrow Z_4$	001	010	- 0	1 0 0	0	1	0
$Z_3 \rightarrow Z_4$	011	010	- -	0 1 0	0	1	0
$Z_4 \rightarrow Z_3$	010	011	0 1	0 0 1	0	1	1
$Z_4 \rightarrow Z_4$	010	010	0 0	0 0 1	0	1	0
$Z_4 \rightarrow Z_5$	010	110	1 -	0 0 1	1	1	0

3.5 Мінімізація функцій тригерів

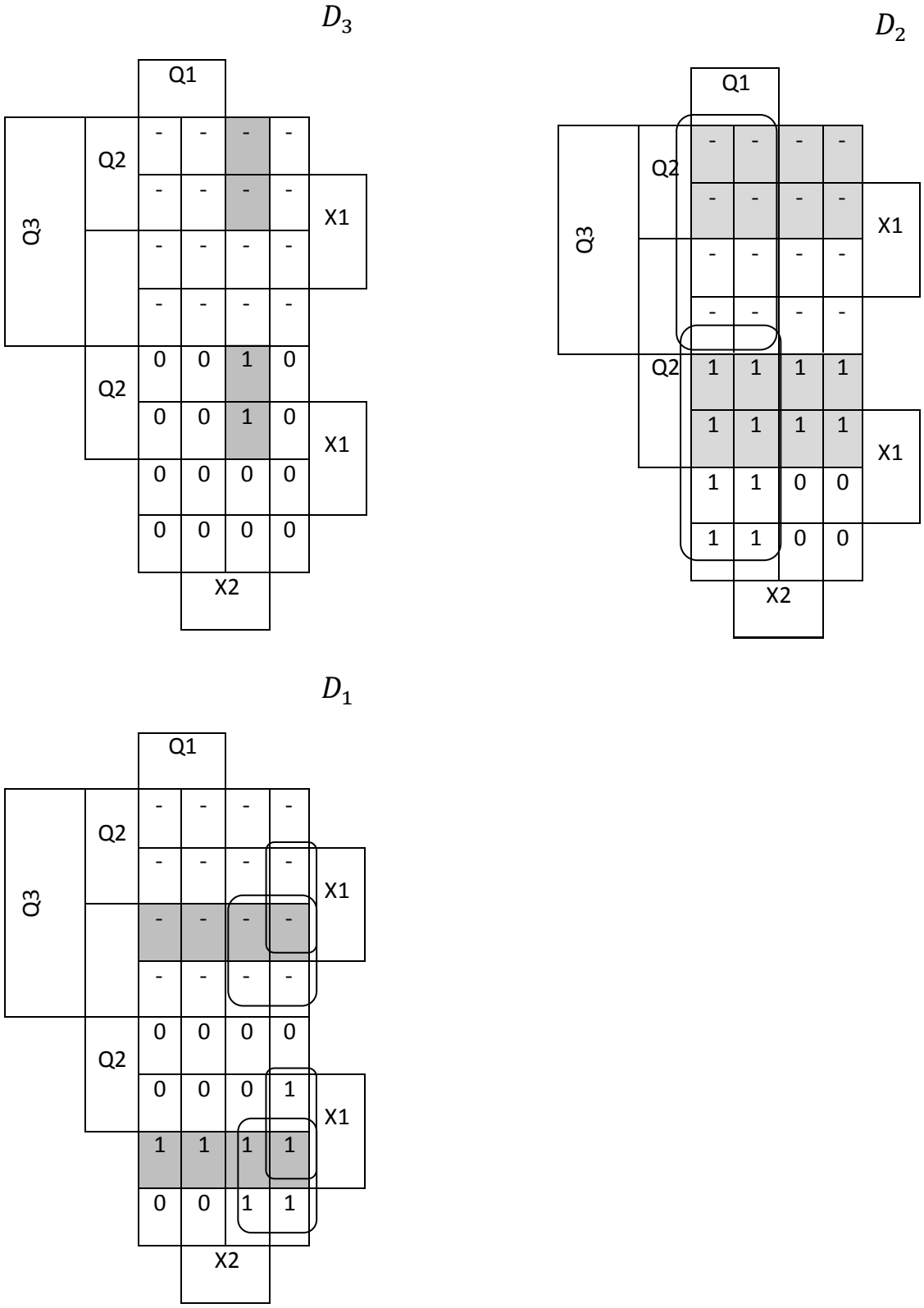

Рисунок 3.3 – Мінімізація функцій тригерів

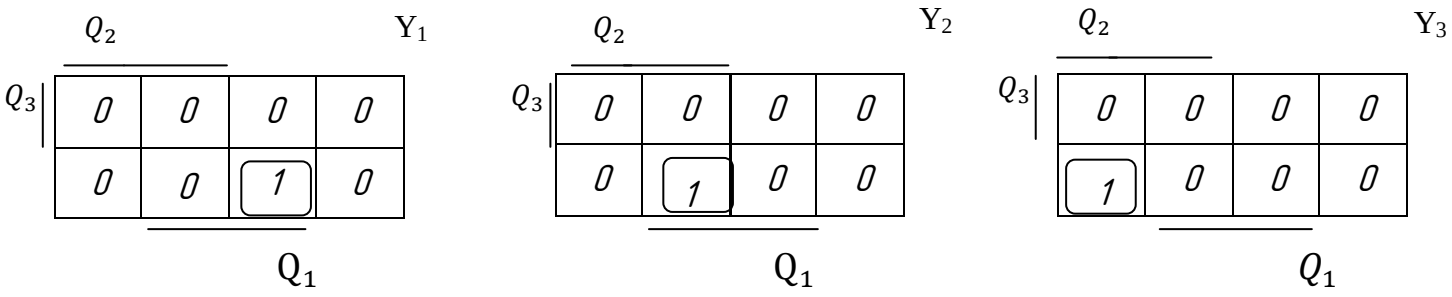

Рисунок 3.4 – Діаграми Вейча для вихідних сигналів

$$D_1 = \overline{Q_2}\ \overline{Q_1} \vee \overline{Q_2} X_1 \vee \overline{Q_1}\ \overline{X_2} X_1$$

$$D_2 = Q_2 \vee Q_1$$

$$D_3 = Q_2 \overline{Q_1} X_2$$

$$Y1 = \overline{Q_3}\ \overline{Q_2} Q_1$$

$$Y2 = \overline{Q_3} Q_2 Q_1$$

$$Y3 = \overline{Q_3} Q_2 \overline{Q_1}$$

3.6 Функціональна схема автомата

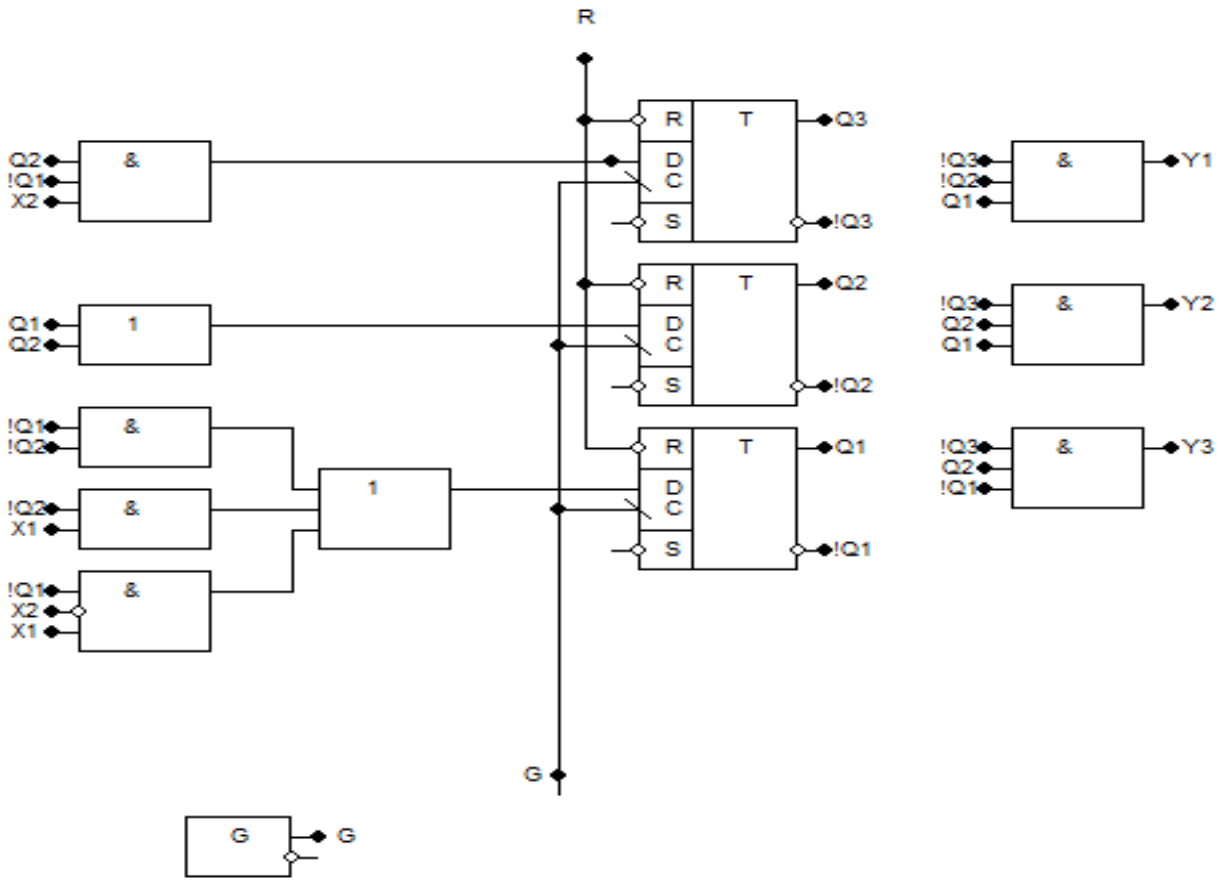

Рисунок 3.5 - Функціональна схема

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 віднімання. Для операції множення першим способом було побудовано управляючий автомат Мура на D-тригерах і елементах булевого базису. Зроблено мінімізацію функцій тригерів і в середовищі AFDK побудована функціональна схема автомата. На одній з функцій використано фільтр для запобігання виникненню просічок.

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»