**Министерство образования Российской Федерации**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. Н.Э. БАУМАНА**

Факультет: Информатика и системы управления

Кафедра: Информационная безопасность (ИУ8)

**АППАРАТНЫЕ СРЕДСТВА**

**ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

**Рубежный контроль №3 на тему:**

«Синтез специализированного процессора»

Вариант 07

**Преподаватель:**

Рафиков А.Г.

**Студент**:

Кудрявцев И.А.

**Группа:**

ИУ8-65

Москва 2020

**Оглавление**

Оглавление

[Цель и задачи работы 3](#_Toc42972027)

[Теоретическая часть 3](#_Toc42972028)

[Практическая часть 4](#_Toc42972029)

[Описание задачи для выполнения (вариант 07) 4](#_Toc42972030)

[Необходимые теоретические выкладки 5](#_Toc42972031)

[Реализация алгоритма преобразования двоичного кода в унитарный 6](#_Toc42972032)

[Алгоритм из СР5(РК2) 7](#_Toc42972033)

[Таблицы заполнения управляющей памяти 8](#_Toc42972034)

[Формат микрокоманд 9](#_Toc42972035)

[Прошивка ПЗУ 10](#_Toc42972036)

[Описание принципа работы процессора 11](#_Toc42972037)

[Схема модели процессора, собранного в Proteus 12](#_Toc42972038)

[Определение быстродействия разработанного процессора (защита ЛР1) 13](#_Toc42972039)

[Результат работы процессора 14](#_Toc42972040)

[Вывод 14](#_Toc42972041)

# Цель и задачи работы

Разработать модель специализированного процессора, для этого синтезировать управляющий и операционный автоматы, собрать специализированный. Процессор, для которого реализовать систему команд, на основе заданных алгоритмов и выполнить в составе СУЦВМ последовательность обязательных команд, каждая из которых реализует заданный специализированный микропрограммный алгоритм и вспомогательных команд, реализующих штатные алгоритмы операций процессора.

# Теоретическая часть

При описании узлов и устройств цифровой обработки их часто представляют в виде композиции управляющей и операционной частей или управляющего и операционного автоматов. Операционный автомат (ОА) выполняет конкретные операции преобразования информации (шифраторы, дешифраторы, регистры и т. д.). Функцией управляющего автомата (УА) является координация работы операционных устройств. Задача УА — выработка распределенной во времени последовательности сигналов, определяющих порядок работы операционного автомата.

Любая операция, выполняемая ОА, может быть представлена совокупностью микроопераций. Для реализации команды, операции или процедуры (микропрограммы) необходимо на соответствующие управляющие входы операционного автомата подать определенным образом распределенную во времени последовательность управляющих функциональных сигналов.

УА генерирует последовательность управляющих сигналов, предписанную микропрограммой и соответствующую значениям логических условий. Функция управляющего автомата определяется:

* множеством входных осведомительных сигналов, отображающих состояние операционного автомата;
* множеством выходных (управляющих) сигналов, инициирующих выполнение микроопераций, реализуемых операционным автоматом;
* закодированной граф-схемой алгоритма (ГСА), задающей порядок следования управляющих сигналов в зависимости от значений осведомительных сигналов.

Процедура построения УА с хранимой микропрограммной логикой по имеющейся ГСА заключается в следующем:

1. Выбирают способ адресации и формат микрокоманд.
2. Производят разметку ГСА в соответствии с правилами, которые определяются выбранным способом адресации.
3. Составляют кодированную микропрограмму в виде таблицы, строки которой соответствуют отметкам на ГСА.
4. Выбирают типы необходимых микросхем и составляют структурную и принципиальную схемы автомата.

# Практическая часть

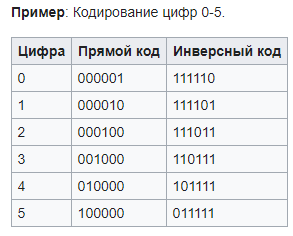
## Описание задачи для выполнения (вариант 07)



* Тип УА (исходя из таблицы выше): МПА с сокращенным тактом
* Алгоритм, который необходимо реализовать для выполнения работы: Микропрограмма для преобразования 4-разрядного двоичного кода в унитарный код (причина выбора 4-разрядного, а не 8-разрядного будет приведена дальше).
* Используемый ОА – К1804ВС1

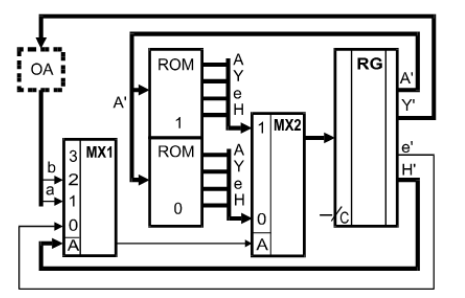
## Необходимые теоретические выкладки

**Унитарный код** — двоичный код фиксированной длины, содержащий только одну 1 — *прямой унитарный код* или только один 0 — *обратный (инверсный) унитарный код*. Длина кода определяется количеством кодируемых объектов, то есть каждому объекту соответствует отдельный разряд кода, а значение кода положением 1 или 0 в кодовое слове.



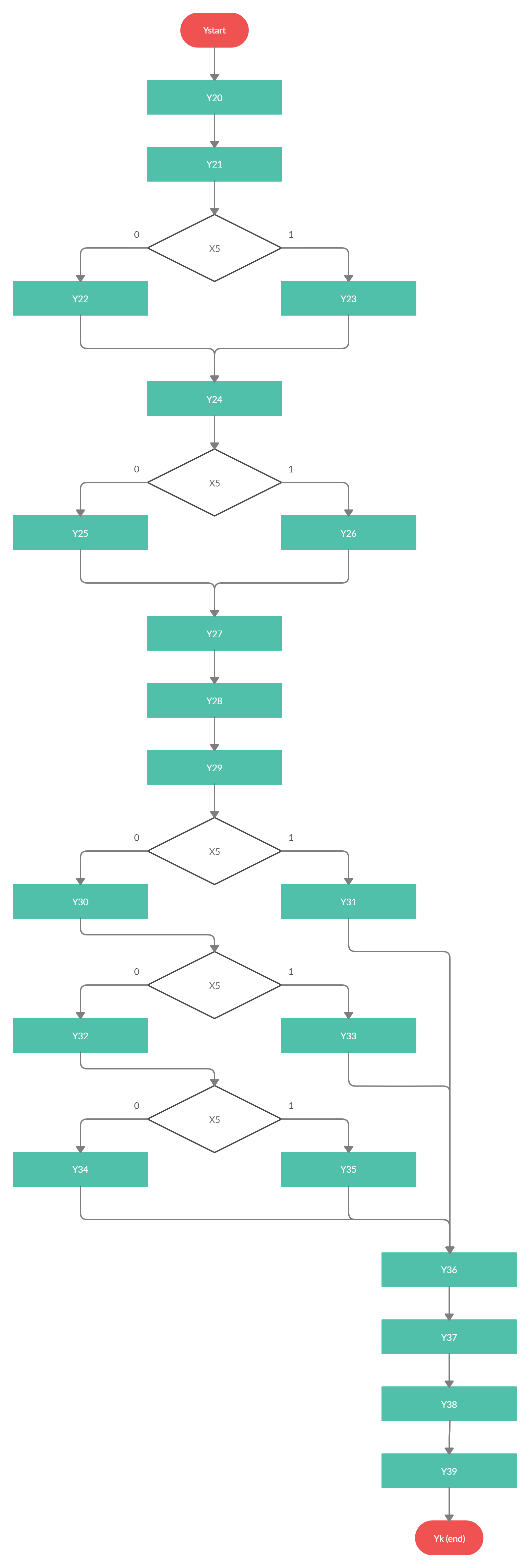
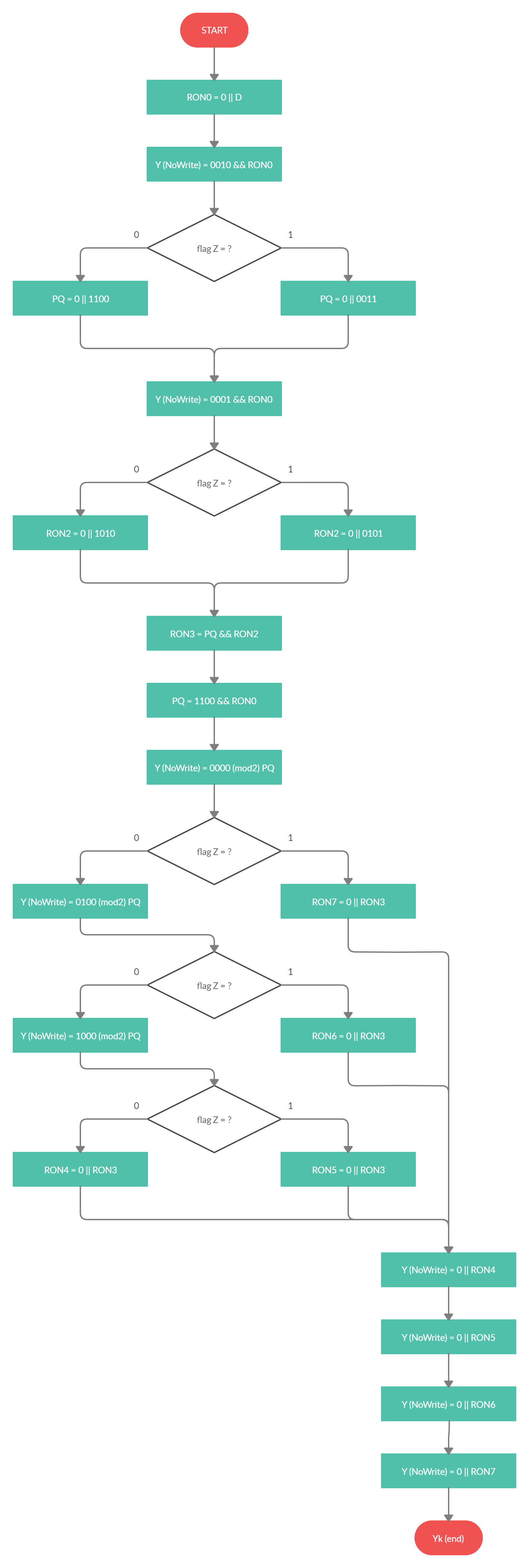
**Исходя из теории следует**, что для кодирования 8-разрядного двоичного кода в унитарный код нужно разрядов (длина унитарного кода). В ОА (при использовании сдвоенного ОА для преобразования 8-разрядного двоичного кода) максимальное число разрядов, которые можно использовать (в каждом ОА содержится 16 РОНов по 4 разряда + PQ в 4 разряда) = . Очевидно, что такого количества разрядов не хватит для записи 256-разрядного унитарного кода, поэтому было принято решение реализовать преобразование 4-разрядного двоичного кода в унитарный код, используя при этом один ОА (для записи унитарного кода нужно уже разрядов, которые имеются в одном ОА).

Схема УА с сокращенным тактом имеет вид:



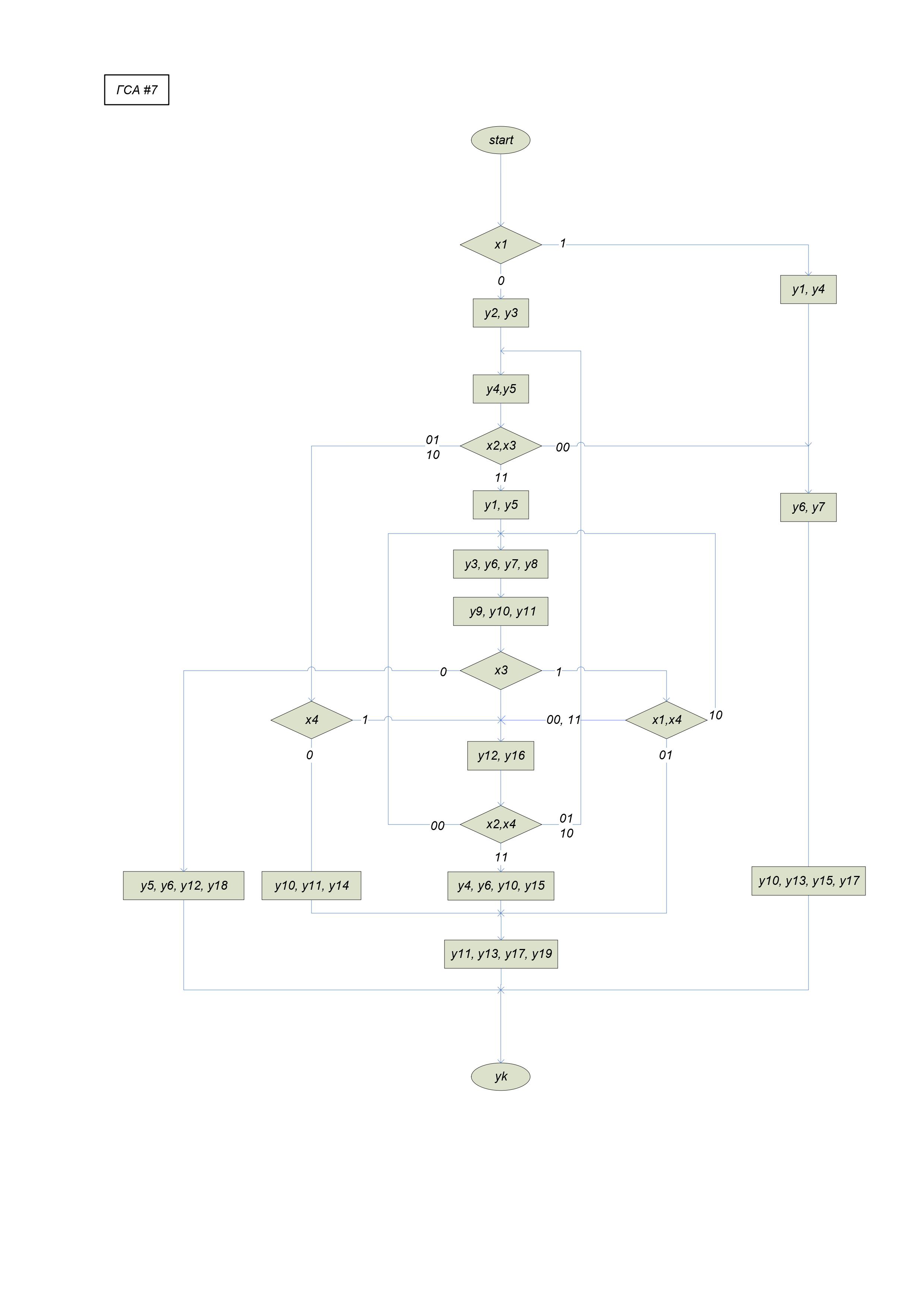
## Реализация алгоритма преобразования двоичного кода в унитарный

**ГСА преобразования 4-разрядного двоичного кода в унитарный код (обычная и закодированная):**



## Алгоритм из СР5(РК2)

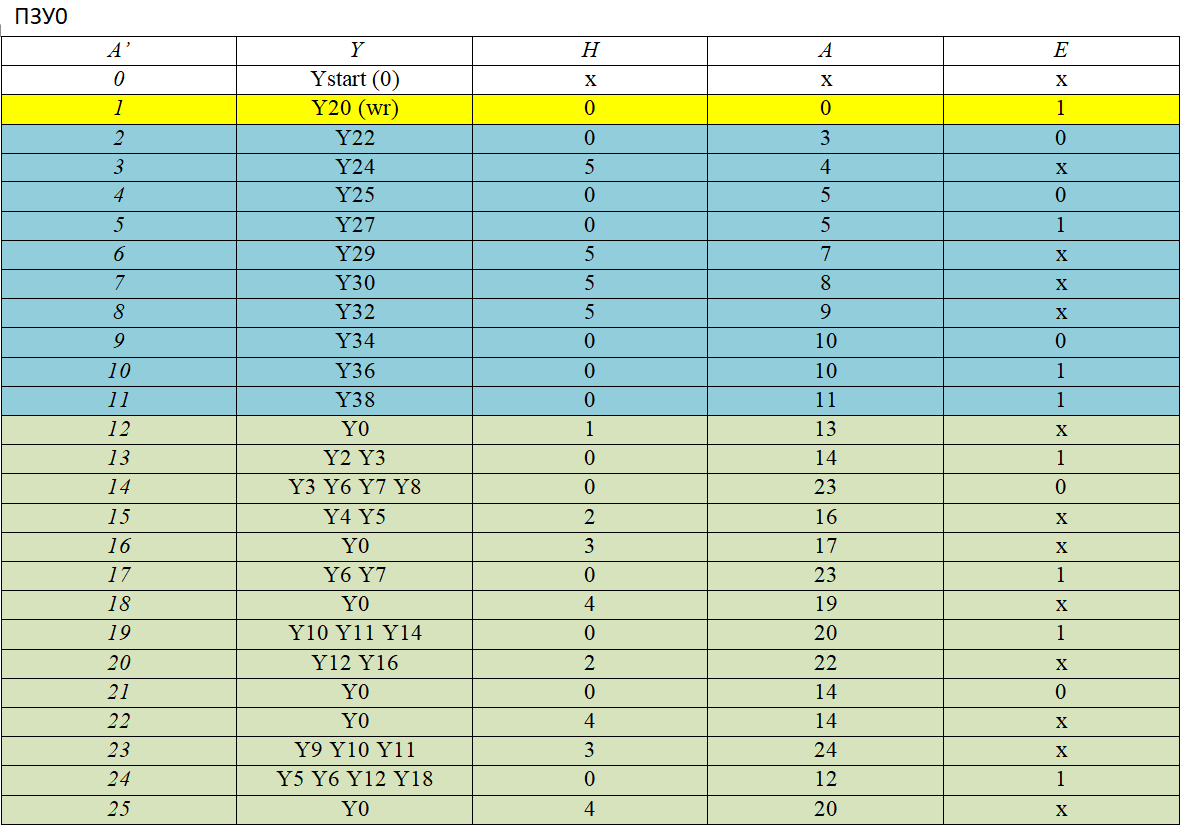
**ГСА алгоритма, из СР5 (РК2)**

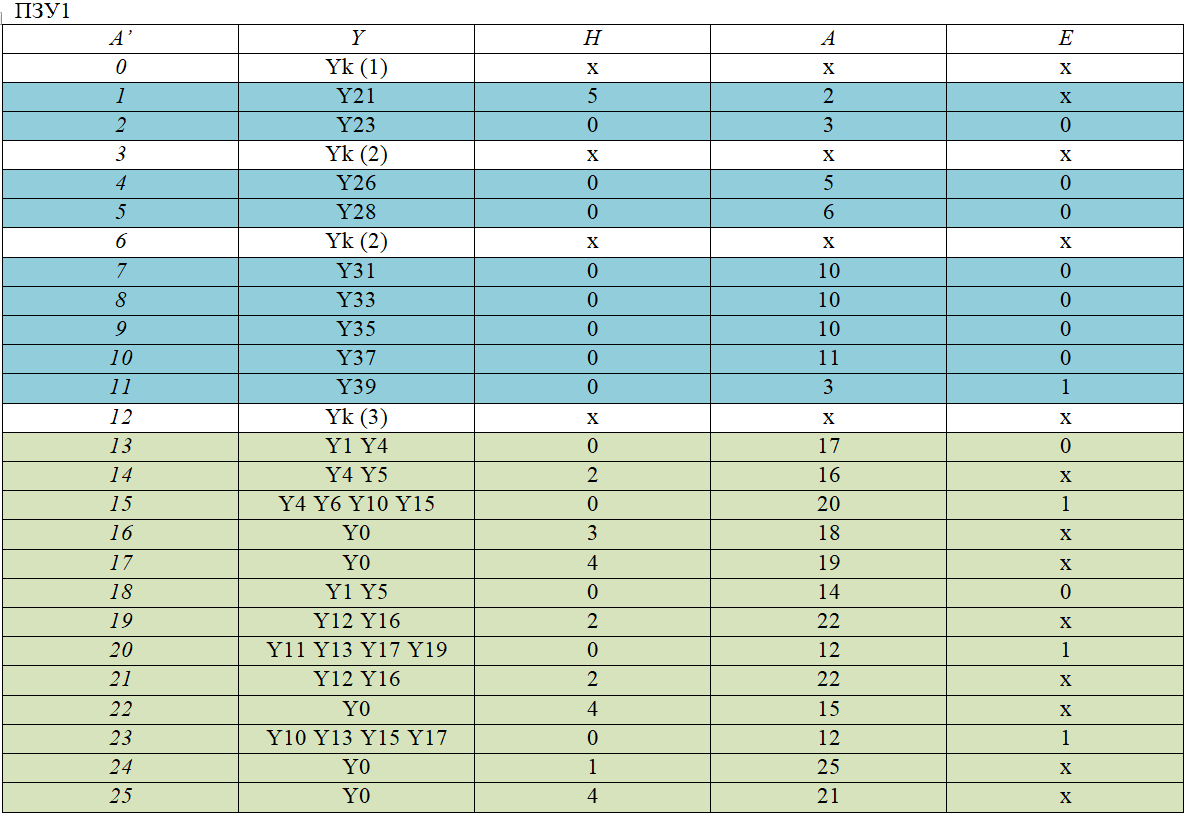


## Таблицы заполнения управляющей памяти

Исходя из **вида УА**, закодированного **ГСА для алгоритма** преобразования двоичного кода в унитарный, а также **ГСА из СР5 (РК2)**, составим заполнение наших ПЗУ0 (ROM0) и ПЗУ1 (ROM1):

**Заполнение ПЗУ0 и ПЗУ1 имеет следующий вид**





* ***Синим*** цветом указаны микрокоманды, относящиеся к команде, реализующей алгоритм преобразования двоичного кода в унитарный
* ***Зеленым*** цветом указаны микрокоманды, относящиеся к команде, реализующей алгоритм, изображенный на ГСА из СР5 (РК2)
* ***Желтым*** цветом указана микрокоманда – вспомогательный алгоритм, для считывания данных с шины данных (ШД)

## Формат микрокоманд

**Определим формат микрокоманд обоих алгоритмов**

1. Формат микрокоманды для алгоритма преобразования двоичного кода в унитарный:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Z***  (Z4-Z0) | ***H***  (H2-H0) | ***e*** | ***I2-0*** | ***I5-3*** | ***I8-6*** | ***A***  (A3-A0) | ***B***  (B3-B0) | ***D***  (D3-D0) | ***K***  (K1-K0) | E |

Где:

* Z4-Z0 – адрес следующей МК в ПЗУ
* H2-H0 – адрес для управления MUX флагов
* e – разряд для выбор одного из ПЗУ МК, если осуществляется безусловный переход
* I2-0, I5-3, I8-6 – информация об операции для ОА
* A3-A0, B3-B0, D3-D0 – данные (прописанные в ПЗУ МК) для ОА
* K1-K0 – адрес команды в ПЗУ команд
* E – разряд, для разрешения начала выполнения команды из ПЗУ команд

1. Формат микрокоманды для ГСА из СР5 (РК2):

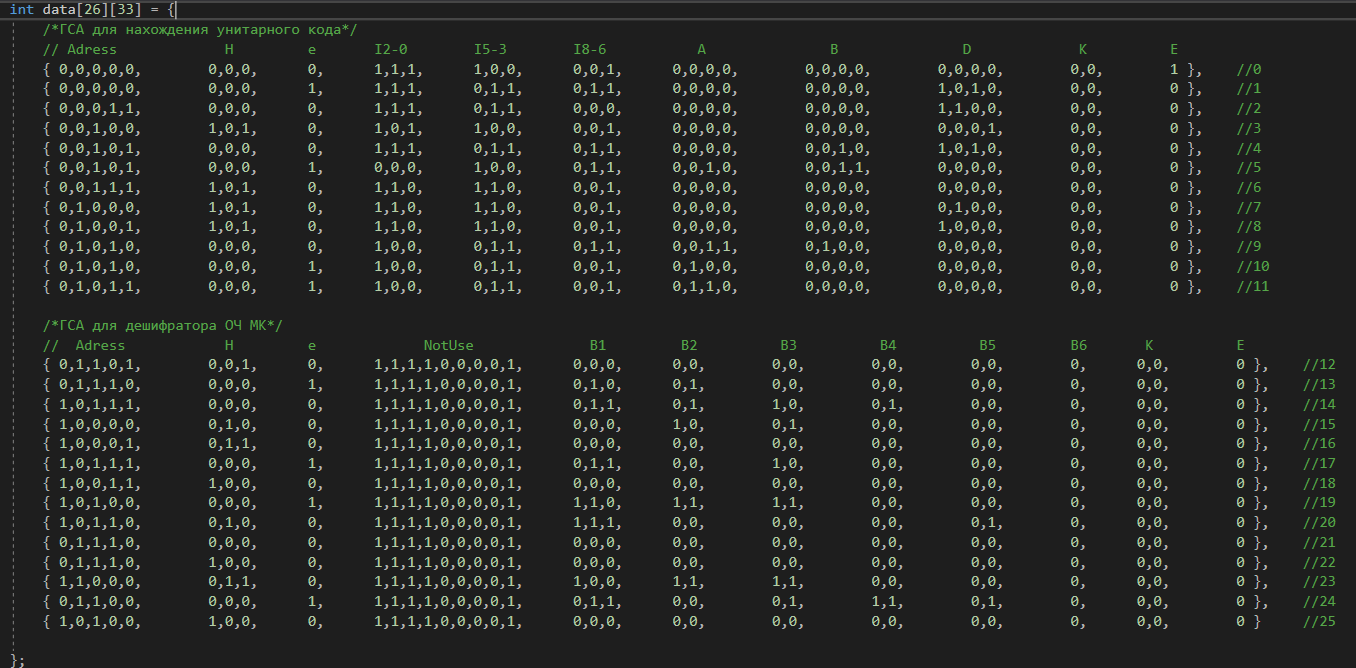
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Z***  (Z4-Z0) | ***H***  (H2-H0) | ***e*** | - | - | - | ***A***  (A3-A0) | ***B***  (B3-B0) | ***D***  (D3-D0) | ***K***  (K1-K0) | E |

Где:

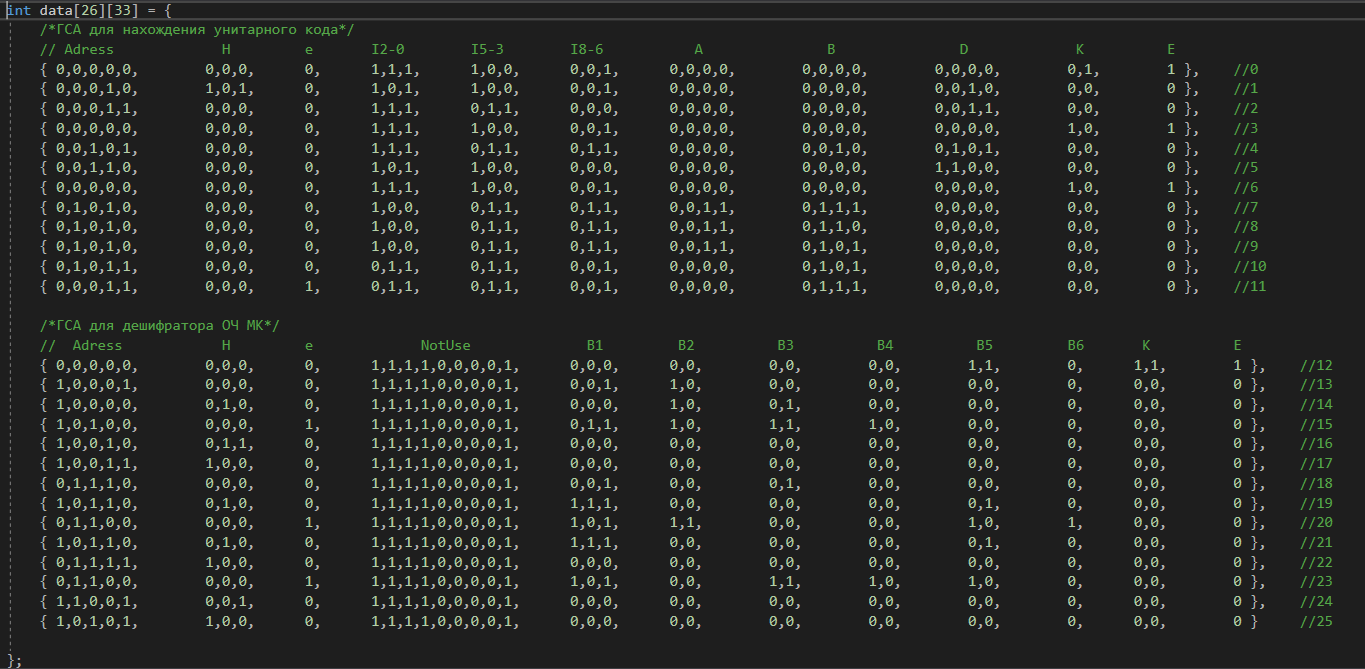
* Z4-Z0 – адрес следующей МК в ПЗУ
* H2-H0 – адрес для управления MUX флагов
* e – разряд для выбор одного из ПЗУ МК, если осуществляется безусловный переход
* A3-A0, B3-B0, D3-D0 – данные (прописанные в ПЗУ МК) для дешифратора ОЧ МК (задача СР5( РК2) )
* K1-K0 – адрес команды в ПЗУ команд
* E – разряд, для разрешения начала выполнения команды из ПЗУ команд

## Прошивка ПЗУ

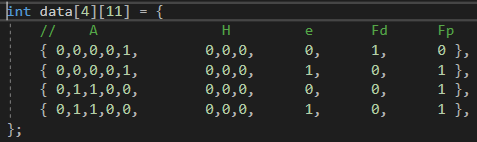
1. Прошивка ПЗУ0 МикроКоманд:



1. Прошивка ПЗ1 МикроКоманд:



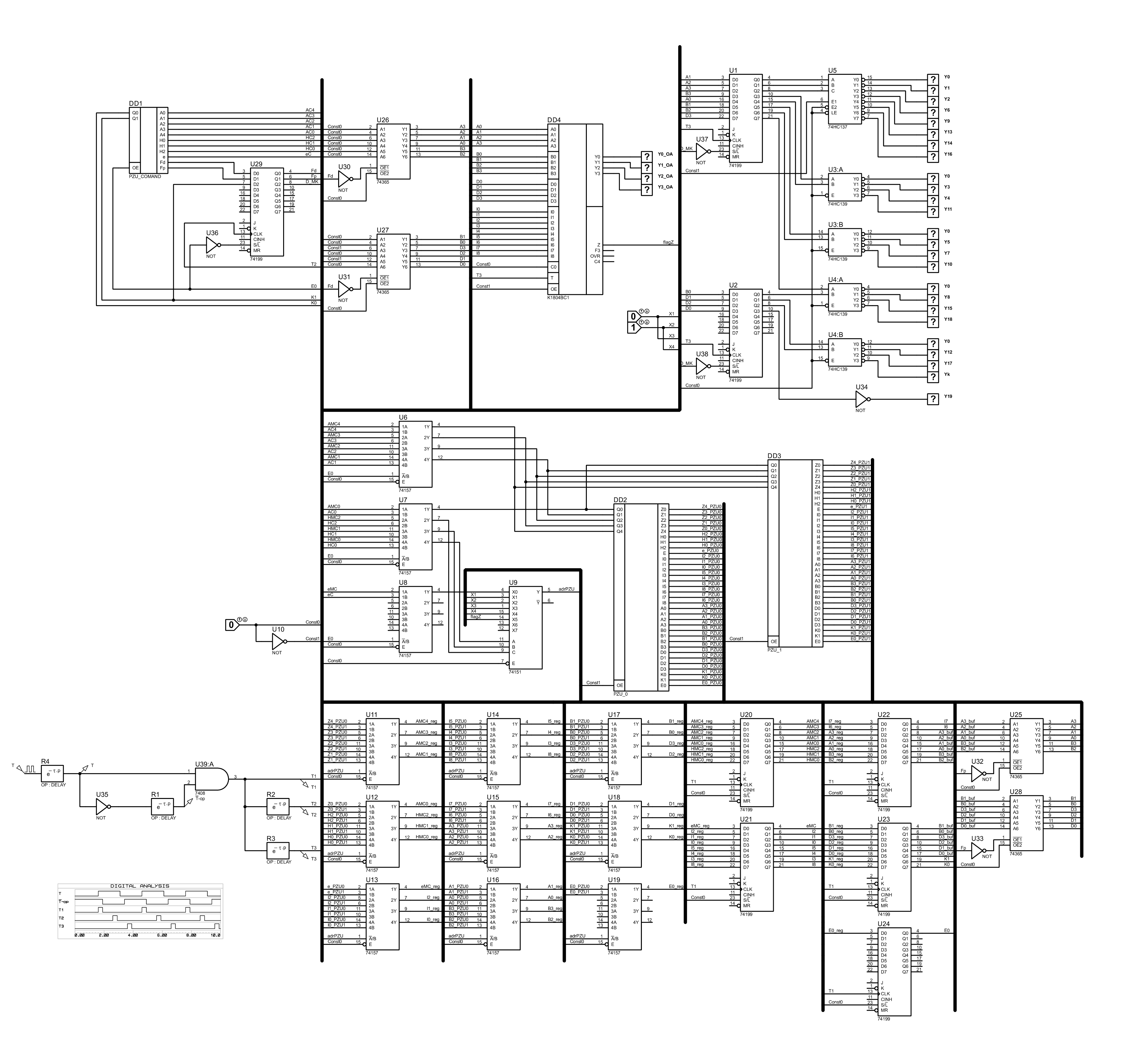
1. Прошивка ПЗУ Команд:



## Описание принципа работы процессора

Процессор начинает работу с 0 МК ПЗУ0, в этой МК записана информация для ПЗУ команд. ПЗУ команд дает команду выполнения записи данных в ОА в РОН0 с ШД (а не с данных ПЗУ МК). Далее следующая микрокоманда дает сигнал ПЗУ команд на подачу следующей команды. Следующая команда – выполнение алгоритма преобразования двоичного кода в унитарный (во время выполнения этой команды данные считываются уже с ПЗУ МК). После выполнения данной команды в ОА по адресам РОН4-РОН7 будет записан унитарный код нашего двоичного числа. Затем следующая микрокоманда содержит сигнал на подачу следующей команды. Следующая команда – выполнение ГСА из СР5 (РК2), также разрешается работа дешифратора ОЧ МК с помощью некоторого флага. После окончания команды работы с ГСА процессор останавливается – вся работа выполнена.

## Схема модели процессора, собранного в Proteus



## Определение быстродействия разработанного процессора (защита ЛР1)

Исходя из схемы, определим быстродействие разработанного процессора. Полный цикл работы процессора состоит из:

1. MUX для выбора A, e, H (взять из ПЗУ команд или ПЗУ МК)
2. MUX выбора флага
3. ПЗУ МК
4. MUX выбора ПЗУ МК
5. Регистры для запоминания данных из ПЗУ МК + буферы данных для ПЗУ МК
6. ПЗУ команд
7. Регистр “флагов” из ПЗУ команд
8. Работа ОА **или** работа дешифратора ОЧ МК (Регистр + дешифратор)

Далее цикл повторяется

Максимальные задержки на этапах:

1. MUX 74157 – 9 нс
2. MUX 74151 – 25 нс
3. ПЗУ МК – ~1 нс (проверил опытным путем)
4. MUX 74157 – 9 нс
5. Register 74199 – 40 нс **и** Buffer 74365 – 16 нс
6. ПЗУ команд – ~1 нс (проверил опытным путем)
7. Register 74199 – 40 нс
8. ОА – ~125 нс

(Register 74199 – 40 нс + Дешифратор – ~20 нс, *но это меньше*)

***Таким образом: 9 + 25 + 1 + 9 + (40 + 16) + 1 + 40 + 125 = 266 нс***

## Результат работы процессора

В результате работы процессора и обрабатывания им 2 команды (то есть алгоритма преобразования двоичного кода в унитарный) в РОН4-РОН7 будет записан верный результат, **например**:

Входной двоичный код – 1010

Выходной унитарный код – 0000 0100 0000 0000

Далее будет выполняться 3 команда ПЗУ команд (то есть ГСА из СР5 (РК2) ), в процессе работы с этой командой на выходах дешифраторов будут появляться нужные сигналы (инверсные).

Далее процессор остановится.

# Вывод

В результате выполнения данной работы была изучена адресация с сокращенным тактом. На основе данной МПА с СТ был построен процессор для выполнения двух простых задач – преобразование двоичного кода в унитарный а также ГСА из СР5 (РК2). Схема процессора была построена в среде Proteus и промоделирована там. Результат работы процессора оказался верным. Также было определено быстродействие разработанного процессора.