Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

МЕТОДИЧНІ ВКАЗІВКИ
ДО КУРСОВОГО ПРОЕКТУВАННЯ
з дисципліни
ТЕОРІЯ ПРОЕКТУВАННЯ ЕОМ

Одеса 2024

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

МЕТОДИЧНІ ВКАЗІВКИ
ДО КУРСОВОГО ПРОЕКТУВАННЯ
з дисципліни
ТЕОРІЯ ПРОЕКТУВАННЯ ЕОМ

Затверджено
на засіданні кафедри комп'ютерних
інтелектуальних систем та мереж
Протокол № 8 від 25.01.2024.

Одеса 2024

Методичні вказівки до курсового проектування з дисципліни «Теорія проектування ЕОМ» для здобувачів спеціальності 123 – "Комп’ютерна інженерія", освітня програма "Комп’ютерні системи та мережі". /Укл.: І.Г.Мілейко, Є.В. Шендрик, І.П. Блінов – Одеса: Національний університет «Одеська політехніка», 2024. – 30с.

Укладач:
І.Г. Мілейко, к.т.н., доцент
Є.В. Шендрик, к.т.н., доцент
І.П. Блінов, ст. викладач

ЗМІСТ

ВСТУП

Об'єктом курсового проектування по дисципліні " Теорія проектування ЕОМ " є процесор, що виконує заданий обмежений набір команд із системи команд процесора ЄС ЕОМ (IBM/370). Вибір цієї системи команд обумовлений її чіткістю, конкретністю та розвинутою системою адресації. Розробка процесора здійснюється в наступних аспектах:

- написання мікропрограм алгоритмів виконання кожної команди;
- написання об'єднаної змістовної мікропрограми роботи процесора;
- синтез структури операційного автомата процесора;
- синтез керуючого автомата із програмувальною логікою.

Дані методичні вказівки містять у собі короткий опис структурної і функціональної організації проектуємого процесора на рівні основних компонентів структури ЕОМ і їхнього зв'язку з процесором, опис принципів адресації інформації і процедур звертання до основної і регістрової пам'яті, а також форматів команд і даних, використовуємих при розробці процесора. Приводиться опис порядку виконання операцій, реалізованих процесором, і принципу реалізації всіх етапів виконання операцій, до яких входять: вибірка команди; формування адрес і вибірка операндів; обробка операндів; запис результату; формування ознаки результату; дії, зв'язані з обробкою програмних переривань.

Приводяться посилання на відповідні джерела, у яких дається докладний опис кожної команди.

1 ФУНКЦІОНАЛЬНА ОРГАНІЗАЦІЯ ПРОЕКТУЄМОГО ПРОЦЕСОРА

1.1 Структура ЕОМ

Типова структура ЕОМ, до складу якої повинен увійти проектуємий процесор, наведена на рис.1. Вона складається з процесора, що поєднує в собі арифметичний пристрій і центральний пристрій керування, основної (оперативної) пам'яті (**ОП**), регістрової (надоперативної) пам'яті (**РП**) і процесорів вводу-виводу. Зовнішні пристрої підключаються до процесора й **ОП** через процесори вводу-виводу. Докладно функціональна організація ЕОМ цього типу описана в [4].

Рисунок1 - Типова зпрощена структурна схема ЕОМ.

Процесор вибирає з **ОП** команди і виконує всі операції, за винятком операцій вводу-виводу. **ОП** служить для прийому і видачі інформації (операндів і команд), що беруть участь у найближчому ряді операцій. **РП** використовується для збільшення швидкодії процесора.

Основна пам'ять забезпечує збереження байтів інформації, граничне число яких (гранична ємність **ОП**) дорівнює 2^{24} = I67772I6 байтів. Різні моделі ЕОМ можуть мати різну ємність **ОП**. Більш продуктивні моделі комплектуються пам'яттю більшої ємності. За одне звертання до **ОП** записується або читається кілька байтів інформації: 2, 4 чи 8 байт.

Позначимо: **ОП[0:N](0:n)** - двовимірний масив битів інформації, збережених в основній пам'яті, що складається з **N + 1** рядків і з **n** стовпців (розрядів).

Регістрова пам'ять (**РП**) процесора складається з 16 регистрів загального призначення **РЗП[0:15](0:31)**, кожний з який забезпечує збереження 32-розрядного слова, і з чотирьох

регистрів з комою, що **плаває, РПК[k](0:63)**, де **k** = 0, 2, 4, 6. Кожен **РПК** забезпечує збереження 64-розрядного слова. **РЗП** идентифікуються адресами 0, 1,..., 15; а **РПК** - адресами 0, 2, 4, 6.

Кожному **РЗП** відповідає одна комірка **РП**, кожному **РПК** - дві комірки **РП**, тому **РП** являє собою двовимірний масив битів інформації, що складаються з 24 рядків (слів) і з 32 стовпців (розрядів).

Вкажемо еквівалентні позначення регістрової пам'яті процесора:

РЗП[0.15] = РП[0:15],
РПК[0] = РП[16].РП[17],
РПК[2] = РП[18].РП[19],
РПК[4] = РП[20].РП[2l],
РПК[б] = РП[22].РП[23].

Крапка позначає з'єднання двох рядків (слів) масиву **РП** для одержання подвійного слова довжиною 64 розряди.

Для ідентифікації слів регістрової пам'яті потрібна адреса довжиною п'ять двійкових розрядів. Уведемо відповідні позначення для **РП**, **РЗП** і **РПК**, де в правій частині адреси комірок регістрової пам'яті, зазначені в двійковій системі числення.

РП[0:23] = РП[00000:10111],
РЗП[0:15] = РП[00000:01111],
РПК[0] = РП[10000]. РП[10001],
РПК[2] = РП[10010]. РП[10011],
РПК[4] = РП[10100]. РП[10101],
РПК[6] = РП[10110]. РП[10111].

При звертанні до **РЗП** значення старшого розряду двійкової адреси дорівнює нулю, при звертанні до **РПК** – одиниці. Кожен **РПК** складається з двох слів з парною і непарною адресою. Адреса **РПК** збігається з адресою парного слова **РП**, тому прийнята нумерувати **РПК** парними числами.

1.2 Розміщення й адресація інформації в ЕОМ на базі проектуємого процесора.

1.2.1 Машинні елементи інформації.

У розглянутій ЕОМ основним елементом інформації, що адресується, передається та оброблюється як одне ціле, є байт.

Двійкові розряди байта нумеруються зліва направо значеннями 0, 1,…7... На основі байтів будуються наступні елементи: ***півслова***, що складаються з двох байтів; ***слова*** з чотирьох байтів; ***подвійні слова*** з восьми байтів; ***поля змінної довжини***, що можуть поєднувати в собі від 1 до 256 байтів. Байт, півслово, слово та подвійне слово називаються полями фіксованої довжини. Тип елементу інформації вказується кодом операції. Коди операцій, що виконуються над елементами фіксованої довжини, одночасно обумовлюють і довжину операндів, що може дорівнювати 1, 2, 4 або 8 байтам. Команди, що ініціюють операції над полями змінної довжини, містять у собі поля, за допомогою яких визначається довжина операндів.

1.2.2 Адресація інформації.

Інформація, збережена в **ОП**, идентифікується з точністю до байта адресами 0,1,…2^K-1, де 2^K – ємність **ОП**. Кожен номер визначає деякий байт. Адреса елемента інформації, що складається з декількох байтів, визначається адресою самого лівого байта. Адреса, що виходить за межі фактичної ємності **ОП** конкретної моделі ЕОМ, розглядається як неправильна. У випадку появи неправильної адреси, що може бути результатом помилки в програмуванні, виконання програми припиняється.

Поля фіксованої довжини повинні розміщатися в **ОП**, починаючи з цілочисельній границі, тобто вони повинні мати адресу, кратну числу байтів у полі. Так, адреса байта може

мати довільне значення, оскільки будь-яке число кратне одиниці; адреса півслова повинна бути парною; адреса слова – кратна 4, а адреса подвійного слова – 8 (рис. 2). Адреса елементу вважається специфікованою неправильно, якщо в його двійковому представленні **m** молодших розрядів не дорівнюють нулю, де **m** = $\log_2$**L**; **L** – довжина елемента в байтах. Якщо в команді з'являється неправильно специфікована адреса, то формується сигнал переривання і звертання до пам'яті блокується (неправильна специфікація адреси).

Рисунок 2 - Двійкові адреси одиниць інформації.

Поля змінної довжини можуть починатися з будь-якого байта, тобто вони можуть мати довільну адресу в межах фактичної ємності **ОП**.

1.2.3.Формати даних.

У проектуємому процесорі повинні представлятися й оброблятися дані наступних типів: цілі двійкові числа, числа з плаваючою комою, десяткові числа та логічні значення.

Цілі двійкові числа можуть представлятися в короткому або довгому форматах (рис. 3). Нульовий розряд є знаковим: знак «+» кодується як 0, знак мінус «-» - як 1. Негативні числа зберігаються в пам'яті в додатковому коді.

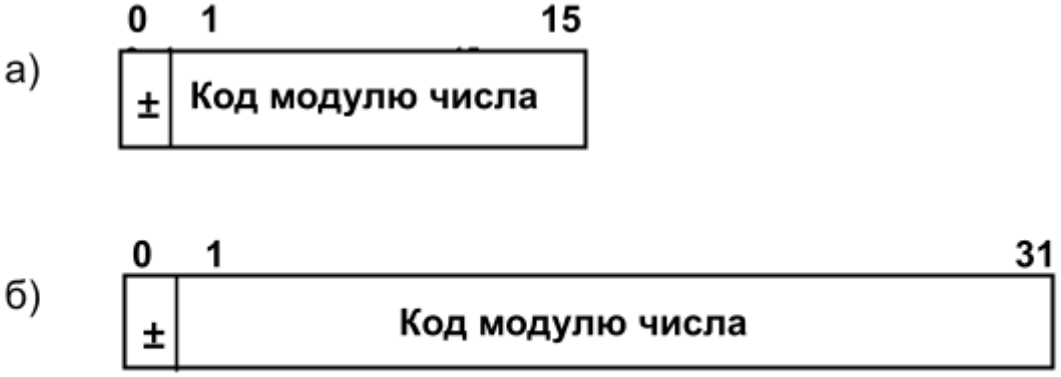

Рисунок 3 - Формати цілих двійкових чисел: а) короткий; б) довгий

Числа з плаваючою комою можуть представлятися в короткому, довгому та розширеному форматах (рис. 4). Нульовий розряд є знаком мантиси. Мантиса представляється у вигляді шістнадцятирічних цифр. Негативні числа зображуються в прямому коді. Характеристика **X** дорівнює порядку числа, збільшеному на 64, і представляє значення порядку в діапазоні від -64 до +63.

Десяткові числа представляються полями змінної довжини двох форматів: у форматі з зоною й в упакованому форматі (рис.5).

У зонному форматі 4 молодших бити байта називаються числовими і звичайно містять код, що представляє десяткову цифру (**D**). Цифра зображується двійковим кодом 0000, 0001, …,1001... Старші 4 бити називаються зоною (**Z**). Зона зображується двійковим кодом 1111 чи 0101.

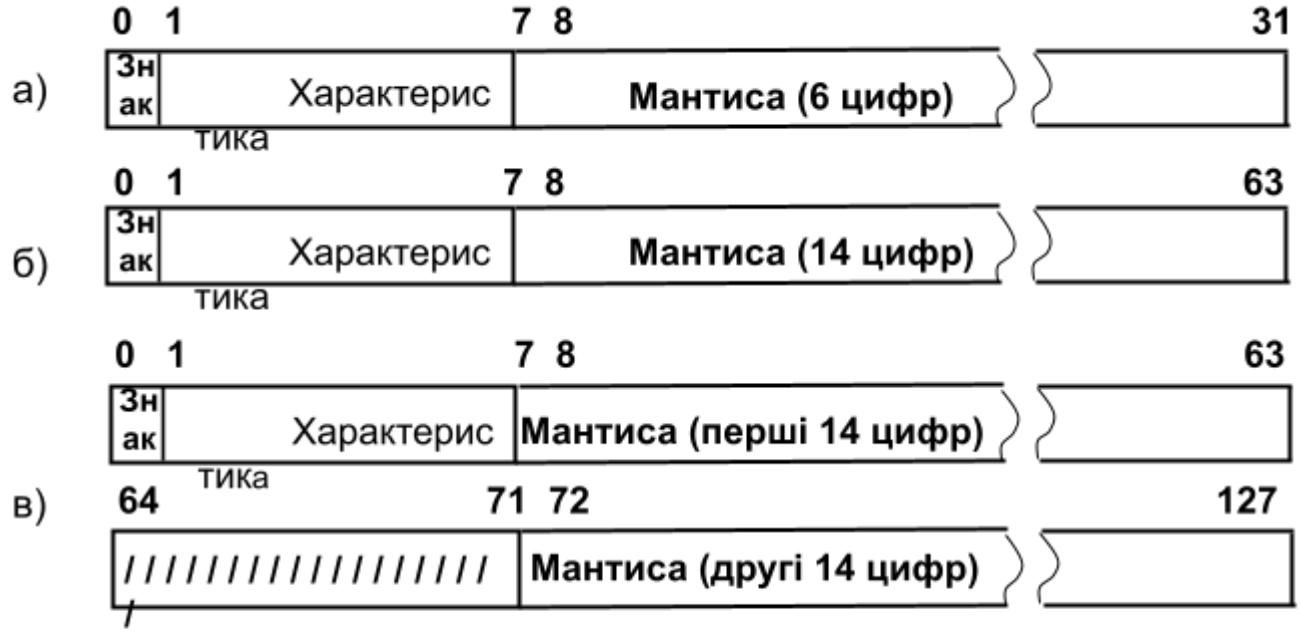

Рисунок 4 - Формати чисел с плаваючею комою: а) короткий, б) довгий, в) розширений

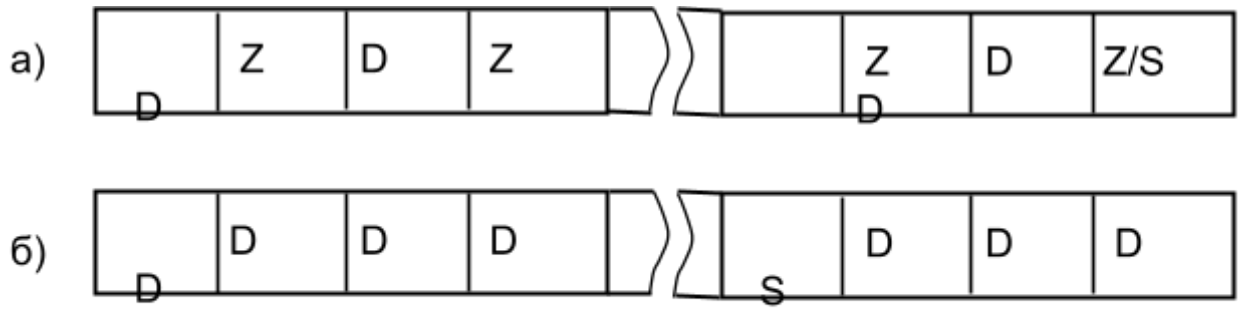

Рисунок 5 - Формати представлення десяткових чисел: а) зонний, б) упакований

Виняток становить самий правий байт поля, у якому старші 4 бити можуть представляти або код зони **Z**, або код знака числа (**S**). Знак «+» кодується одною з комбінацій 1010, 1100, 1110, 1111. Знак «-» - комбінаціями 1011, 1101. Десяткові числа мають довжину від 1 до16 байтів.

В упакованому форматі в кожнім байті розташовуються дві десяткові цифри. Виняток складає самий правий байт, у якому праворуч від цифри поміщений знак числа.

Логічні значення представляються в двох форматах: фіксованої довжини і змінної довжини. У форматі фіксованої довжини кожен бит представляє одне логічне значення, і сукупність логічних значень передається й обробляється як одне ціле. Формат змінної довжини складається з 256 символів, кожний з який кодується одним байтом. Сукупність символів (літер,

знаків, цифр) утворить рядок алфавітно-цифрової інформації – рядок тексту. Більш докладно про формати даних [1, 4].

1.3. Формати команд

Дії над операндами, а також їхні адреси задаються командами. Розроблюємий процесор повинен оперувати із командами різної довжини: 2, 4 і 6 байтів. Довжина команди залежить від того, у якій пам'яті (основній чи регістровій) розміщуються операнди. Адреса основної пам'яті вказується з використанням 2 або 2,5 байтів – адресою типу **S**. Регістрова пам'ять адресується 1/2 байта – адресою типу **R**. У проектуємому процесорі використовується п'ять форматів команд, що позначаються **RR, RX, RS, SI, SS** (рис.6).

Перший байт команди містить код операції **КОП**. Перші два розряди коду операції **КОП(0:1)** обумовлюють формат команди: **00 – RR, 01 – RX, 10 – RS, SI, 11 – SS**. У залежності від формату команди операнди вибираються з регістрової або основної пам'яті. Результат операції записується завжди за адресою першого операнда. У форматі **SI** другий операнд вибирається безпосередньо з команди ($\mathbf{I_2}$).

Поля $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$ є адресами відповідно першого, другого і третього операндів, що зберігаються в регістровій пам'яті (**РЗП** або **РПК**). Поля $\mathbf{B_1}$, $\mathbf{B_2}$, $\mathbf{D_1}$, $\mathbf{D_2}$, $\mathbf{X_2}$ містять інформацію про адреси першого і другого операндів, що зберігаються в основній пам'яті (п.3.2 - Формування адреси операндів). Поля $\mathbf{L_1}$, $\mathbf{L_2}$, **L** містять код довжини операндів змінної довжини, що зберігаються в основній пам'яті.

Формат **RR** обуловлює операцію типу регіистр-регістр, у котру вступають операнди, збережені в регістровій пам'яті. Команда **RX** вказує один операнд з регістрової пам'яті, інший – з основної, причому адреса основної пам'яті допускає індексацію – адреса типу **X.** Команда формату **RS** визначає операцію типу регістр-пам'ять, але без індексації другої адреси. У команді типу **SI** один операнд знаходиться в основній пам'яті, а інший – безпосередньо в команді (випадок безпосередньої адресації, що представляється адресою типу **I**). Формат **SS** обуловлює операцію типу пам'ять-пам'ять над полями змінної довжини.

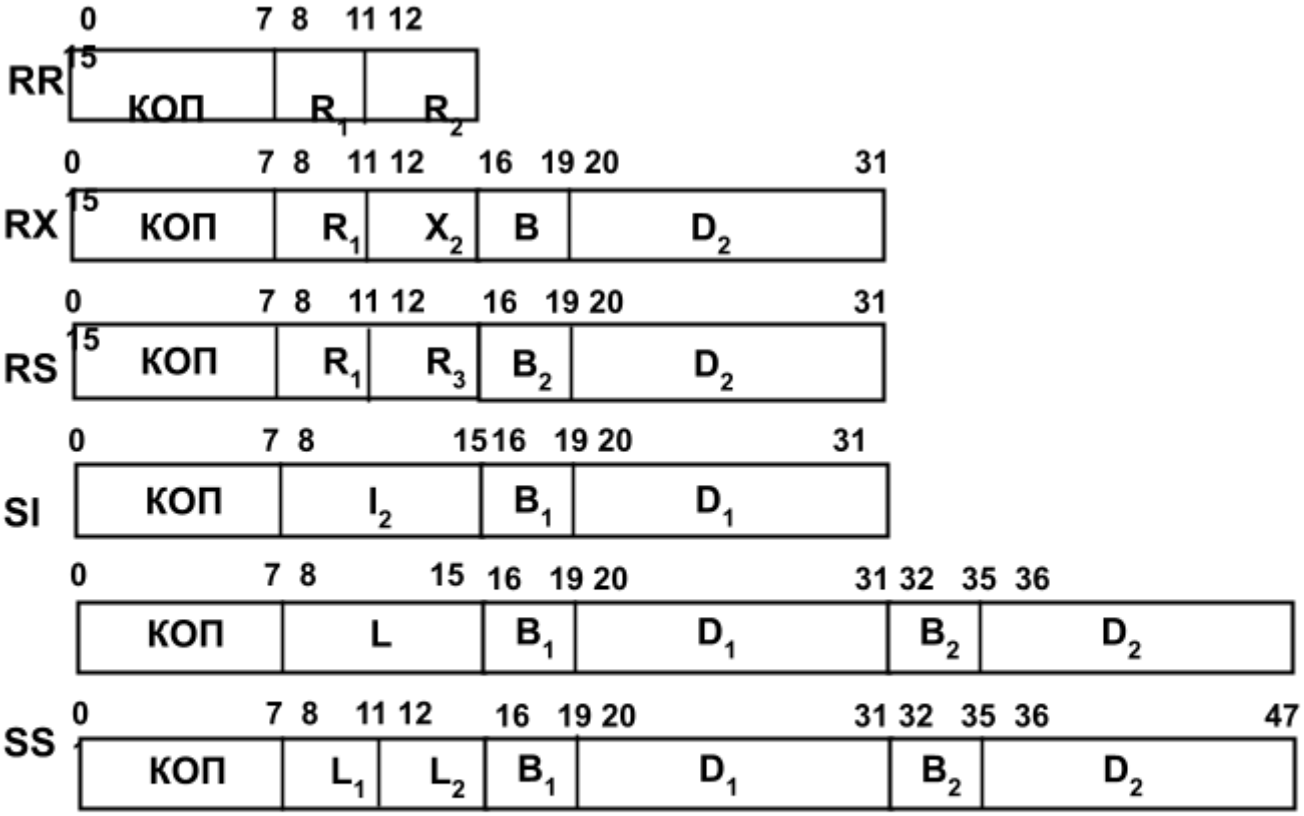

Рисунок 6 - Формати команд проектуємого процессору

Команди різних форматів мають різну довжину: **RR** – півслово; **RX, RS**, **SI** – слово; **SS** – три півслова. Команди будь-якого формату повинні мати адреса, що є цілочисленою границею для півслова (парна адреса).

Усі команди поділяються на наступні основні групи:

- Команди для операцій над числами з фіксованої комою;
- Команди для операцій над числами з плаваючею комою;
- Команди для операцій десяткової арифметики;
- Команди логічної обробки даних;
- Команди переходів.

Команди в 32-розрядної **ОП** можуть розташовуватися у вигляді, представленому на рис.7.

Природно, що для 64-розрядної пам'яті варіантів розташування команд різних форматів набагато більше.

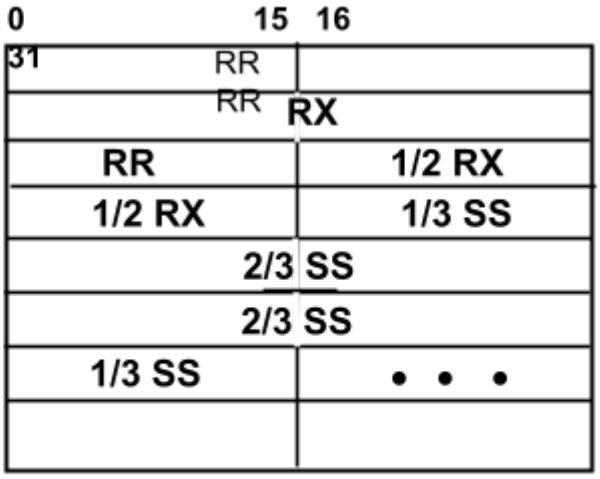

Рисунок 7 - Варіанти розташування команд різноманітних форматів в основній пам'яті

2 ОСНОВНІ ПРИНЦИПИ ВИБІРКИ КОМАНД І ЗВЕРТАННЯ ДО ПАМ'ЯТІ

2.1 Вибірка команд

У проектуємому процесорі використовуються команди довжиною 1, 2, і 3 півслова. Адреса команди (**АК**) повинна бути парною. Вона указує на номер байта, з якого в **ОП** розташовується наступна команда. Довжина наступної команди заздалегідь невідома. Код довжини команди міститься в нульовому і першому розрядах коду операції **КОП(0:1)**: 00 – одне півслово, 01 або 10 – два півслова, 11 – три півслова. Найменша довжина команди – півслово, тому незалежно від формату вибираємої команди, перше півслово необхідно вибрати. Якщо обрана команда **RR** , то вибірка закінчується. Формати **RX, RS, SI** вимагають вибірки другого півслова команди, а **SS** – другого і третього. У форматі **SS** третє півслово можна вибрати на місце другого півслова, якщо попередньо сформувати адресу першого операнда (у цьому випадку поля $\mathbf{B_1}$, $\mathbf{D_1}$ більш не знадобляться). Таким чином, вибірка команд зводиться до завантаження півслова, слова або трьох півслів у залежності від формату обираної команди (табл. 1).

Таблиця 1 - Вибірка команд

Формат	Дія
RR	**КОМ(0:15) := ОП[АК : (АК + 1)]; АК:= АК+2;**
RX, RS, SI	**КОМ(0:31) := ОП[АК : (АК+3)]; АК:=АК+4;**
SS	**КОМ(0:47) := ОП[АК : (АК+5)]; АК:=АК+6;**

2.2 Формування адрес операндів

Спосіб формування адрес операндів обумовлюється форматом команди, що виконується. У форматі **RR** обидва операнди або знаходяться в **РЗП**, якщо **КОП(2)**=0, чи в **РПК**, якщо **КОП(2)**=1. Отже, для формування адреси **РП** для формування старшого розряду, що визначає звертання до **РЗП** і **РПК**, необхідно використовувати другий розряд коду операції **КОП(2)**:

$$\mathbf{A_1:=КОП(2).R_1;\ A_2:=КОП(2).R_2}.$$

У форматі **RX** другий операнд знаходиться в **ОП**, а його адреса обчислюється таким способом:

$$\mathbf{A_2:=РЗП[X_2] + РЗП[B_2] + D_2}.$$

У форматі **RS** другий операнд також знаходиться в **ОП**, але на відміну від формату **RX** адреса другого операнда не містить індексу і:

$$\mathbf{A_2:= РЗП[B_2] + D_2}.$$

У форматі **SI** перший операнд знаходиться в **ОП**, а його адреса обчислюється так само, як і адреса другого операнда у форматі **RS:**

$$\mathbf{A_1:= РЗП[B_1] + D_1.}$$

Другий операнд міститься безпосередньо в команді в розрядах (8:15).

У форматі **SS** обидва операнди знаходяться в **ОП**, а їхні адреси обчислюються таким способом:

$$\mathbf{A_1:= РЗП[B_1] + D_1;} \qquad \mathbf{A_2:= РЗП[B_2] + D_2.}$$

Якщо вміст полів команди $\mathbf{B_1}$, $\mathbf{B_2}$ або $\mathbf{X_2}$ дорівнюють нулю, то і відповідна компонента адреси дорівнює нулю.

2.3 Організація звертання до регістрової пам'яті

У регістровій пам'яті зберігаються операнди довжиною 4 і 8 байтів. Операнд довжиною 4 байти займає одне слово регістрової пам'яті, операнд довжиною 8 байтів – два слова. При цьому завжди адреса першого слова парна, адреса другого слова на одиницю більше адреси першого.

Якщо потрібно з **ОП** завантажити в **РП** півслово (короткий формат з фіксованої комою), то операнд ліворуч доповнюється 16-ю розрядами. Значення цих розрядів збігаються зі знаковим розрядом операнду, що завантажується

Операція «Запис у пам'ять півслова» вибирає з **РП** слово, а в основну пам'ять записує праве півслово.

При звертанні до **РЗП** за півсловом або словом будь-яке значення адреси **РЗП** є припустимим, отже, вважається специфікованим правильно. При звертанні до **РЗП** за подвійним словом адреса **РЗП** повинна бути парною (див. п.1.2.2 -Адресація інформації).

При звертанні до **РПК** за словом або за подвійним словом адреса **РПК** повинна бути парною і менше восьми. Правила звертання до регістрової пам'яті зведені в табл. 2, у якій використані наступні позначення: **A(0:4)** – адреса регістрової пам'яті; **(0:k)** – операнд $\mathbf{(k \in \{15, 31, 63\})}$, **C** – ознаку специфікації (якщо **C**=1, то адреса специфікована неправильно).

2.4 Організація звертань до основної пам'яті

Позначимо: **A(0:23)** – адреса основної пам'яті; **(0:k)** – операнд **(k ∈ {15, 31, 63})**; **C** – ознаку специфікації.

При звертанні до основної пам'яті адреса повинна бути кратна числу байтів операнду фіксованої довжини. Поля змінної довжини можуть мати довільне значення адреси. Організація звертань до основної пам'яті ілюструється табл. 3.

Операції над полями змінної довжини виконуються послідовно, байт за байтом, тому читання і запис виконуються також по байтові, починаючи з байту **ОП[A]** і кінчаючи байтом **ОП[A+L],** де **(L+1)** – довжина операнда в байтах.

Таблиця 2 - Звертання до регістрової пам'яті

Формат даних	Операція	Тип регістрової пам'яті			
		РЗП		**РПК**	
		С	Дія	**С**	Дія
Півслово **Z(0:15)**	ЧтОП ЗпОП		**Z := РП[0XXXX](16:31)** **РП[0XXXX] := (16)Z(0).Z**	-	- -
Слово **Z(0:31)**	ЧтОП ЗпОП		**Z := РП[0XXXX]** **РП[0XXXX] := Z**	**A(1)VA(4)** **A(1)VA(4)**	**Z := РП[10XX0]** **РП[10XX0] := Z**
Подвійне слово **Z(0:63)**	ЧтОП ЗпОП	**A(4)** **A(4)**	**Z := РП[0XXX0].РП[0XXX1]** **РП[0XXX0].РП[0XXX1] := Z**	**A(1)VA(4)** **A(1)VA(4)**	**Z := РП[10XX0].РП[10XX1]** **РП[10XX0].РП[10XX1] := Z**

Примітки:

1. Символ **X** означає, що допустимо будь-яке значення (0 або 1) у даному розряді адреси регістрової пам'яті, зазначене в полі A(0:4); старший розряд A(0) не бере участь у визначенні ознаки специфікації, він обумовлює тип пам'яті (РЗП або РПК).
2. Запис **(16)Z(0)** означає **Z(0).Z(0).Z(0). … Z(0).**

Таблиця 3 - Звертання до основної пам'яті

Формат даних	**С**	Операція	Дія
Байт **Z(0:7)**	-	**ЧтОП; ЗпОП**	**Z := ОП[A]** **ОП[A] := Z**
Півслово **Z(0:15)**	**A(23)**	**ЧтОП** **ЗпОП**	**Z := ОП[A : (A+1)]** **ОП[A : (A+1)] := Z**
Слово **Z(0:31)**	**A(22)VA(23)**	**ЧтОП** **ЗпОП**	**Z := ОП[A : (A+3)]** **ОП[A : (A+3)] := Z**
Подвійне слово **Z(0:63)**	**A(21)VA(22)VA(23)**	**ЧтОП** **ЗпОП**	**Z := ОП[A : (A+7)]** **ОП[A : (A+7)] := Z**
Поле змінної довжини **Z(0:7)**	-	**ЧтОП** **ЗпОП**	**Z := ОП[A + i)], 0≤ i ≤L** **ОП[A + i)] := Z, 0≤ i ≤L**

3 СТРУКТУРНА ОРГАНІЗАЦІЯ ЕОМ І ОБРОБКА КОМАНД

3.1 Структурна організація процесора.

Одна з можливих структур процесора та його зв'язок із **РП** і **ОП** представлена на рис. 8. Для передачі інформації між окремими пристроями використовується магістраль **М.**

3.1.1. Процесор

Процесор складається з операційного і керуючого автоматів. Операційний автомат служить для збереження слів інформації, виконання набору мікрооперацій над ними й обчислення логічних умов. Керуючий автомат забезпечує необхідний порядок проходження мікрооперацій на основі заданих мікропрограм.

Наведена структура операційного автомата процесора побудована на основі принципу узагальнення мікрооперацій. Операційний автомат розділяється на дві частин: запам'ятовуючу і комбінаційну. Пам'ять автомата складається з ряду регістрів, довжина яких дорівнює довжині збережених у них слів. Регістри **А1**, **В1**, **С1** довжиною в слово використовуються для виконання арифметичних операцій. Якщо процесор повинен оперувати з подвійними словами, необхідно в пам'ять операційного автомата ввести 32- розрядні регістри **А2, В2, С2**, що разом з регістрами **А1, В1, С1** забезпечують збереження подвійних слів **А, В, С.** Довжина регістру команди **РК** відповідає максимальному формату команди з набору команд, реалізуємих процесором. Лічильник адреси команд **ЛАК** зберігає адресу команди. Тому що адреса команди завжди кратна півслову, то довжина **ЛАК** обумовлюється ємністю **ОП** у півсловах, тобто $\log_2 E - 1$, де **Е** – ємність **ОП** у байтах. Довжина лічильника тактів **ЛТ** обумовлюється максимальним числом тактів, що відводяться для виконання арифметичних операцій.

Буферний регістр **БР** використовується для збереження частини слова **ОП** у процесі вибірки команд з **ОП** і має довжину 1.або 3 півслова при 32 або 64-розрядної **ОП** відповідно. При використанні 64-розрядної **ОП** буферний регістр для спрощення структури операційного автомата рекомендується виносити за межі пам'яті операційного автомата і підключати до магістралі безпосередньо. Запам'ятовуюча частина автомата зв'язана з комбінаційною шинами **А, В** и **С**. На шини **А** и **В** вибираються операнди, що вступають у мікрооперацію, а шина **С** використовується для занесення результатів виконання мікрооперацій у пам'ять автомата. Операційний автомат підключений до магістралі за допомогою двох шин, зв'язаних із регістром **Z**.

Комбінаційна частина автомата служить для виконання набору мікрооперацій над словами **А** и **В** і обчислення значень набору логічних умов **Х**, необхідних для реалізації мікропрограм. Комбінаційна частина містить наступні операційні елементи: формувач кодів **ФК**, комбінаційний додавач **КД**, зсувачь **ЗСВ**, формувач інформаційних сигналів **Ф.** Формувач кодів служить для формування додаткових кодів, констант і виділення полів. Основне призначення комбінаційного додавача складається в обчисленні суми чисел. Крім того, комбінаційний додавач може виконувати бінарні логічні операції, мікрооперації рахунку та передачу слова через додавач без перетворення. Зсувачь служить для виконання мікрооперацій зсування та прямої передачі. На формувачі **Ф** обчислюються значення інформаційних сигналів **Х**, що представляють відносини.

Регістр **Z** використовується для проміжного збереження результату мікрооперації перед передачею його в пам'ять операційного автомата. Завдяки наявності регістру **Z** можна виконувати мікрооперації вигляду **S:=f(S)**, де **S** – слово, збережене в пам'яті автомата.

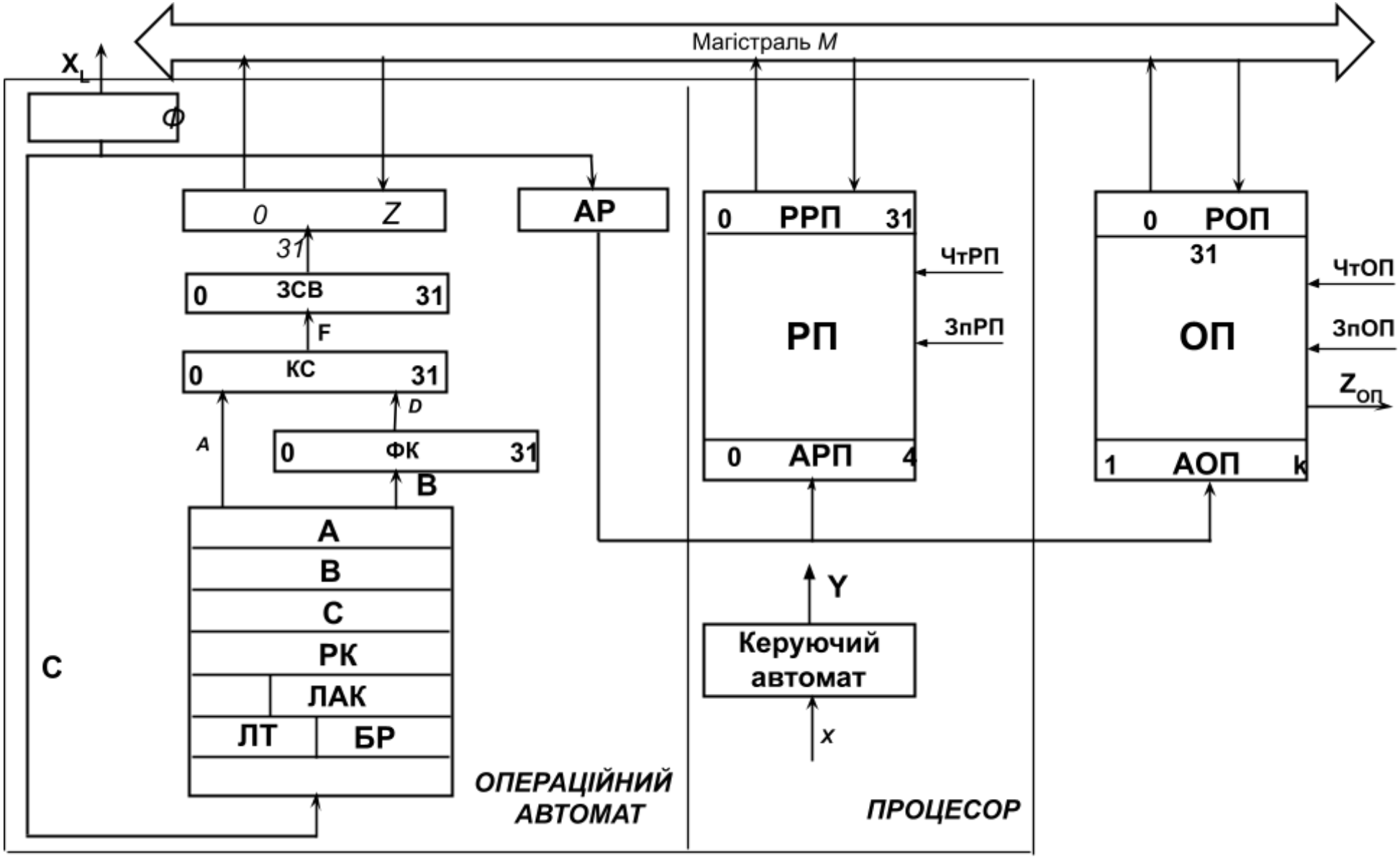

Рисунок 8 - Структурна організація процесора

Для збереження ознак, що відзначають стан процесора доцільно ввести тригери станів. Стани цих тригерів входять у набір інформаційних сигналів **X** операційного автомата. Додатково до складу процесора вводяться регістри, у яких розміщаються поля поточного слову стану програми **ССП** (регістри масок, код довжини команди, ознаки результату, код причини переривання і т.д.). Тригери станів і регістри **ССП** переключаються під впливом сигналів керуючого автомата.

Описана структура процесора є зразковою. Вона повинна бути уточнена і конкретизована студентом при виконанні проекту в залежності від заданого набору команд, особливостей виконання операцій, прийнятих критеріїв оптимальності та інших факторів.

3.1.2 Основна пам'ять. (**ОП**)

У залежності від ширини вибірки **ОП** (32 або 64 бита) за одне звертання до **ОП** читається або записується в **ОП** цілком 32- або 64-розрядне слово. Адреса слова, до якого здійснюється звертання, указується на регістрі адреси основної пам'яті **РАОП**. Довжина регістру **РАОП** дорівнює $\mathbf{log_2 E_C}$ де $\mathbf{E_C}$ – ємність **ОП** у словах, рівна $\mathbf{E_C = E/L}$, **L** – довжина **ОП** у байтах. Слово інформації, що записується або читається з **ОП**, розміщається на регістрі **РОП**. Операція в **ОП** збуджується сигналами читання з основної пам'яті **ЧтОП** і запису в основну пам'ять **ЗпОП**. Момент закінчення операції в **ОП** відзначається сигналом $\mathbf{Z_{оп}}$. Тому що час звертання до основної пам'яті більше тривалості такту роботи процесора, то повинна забезпечуватися синхронізація роботи процесора й **ОП** за рахунок очікуючих вершин графа мікропрограми.

3.1.3 Регістрова пам'ять. (**РП**)

РП застосовується для збільшення швидкодії процесора. **РП** складається з регістрів загального призначення (**РЗП**) і регистрів з плаваючою комою **(РПК**). **РЗП** використовуються як індексні регістри, базові регістри, а також для збереження слів і півслів, що беруть участь в операціях з фіксованою комою. **РЗП** являють собою 32-розрядні регістри й адресуються числами від 0 до 15. Для звертання до **РЗП** у командах будь-якого формату приділяється чотирьох розрядне поле **R**.

При виконанні операцій з плаваючою комою один чи обидва операнди можуть розташовуватися **в РПК**. Усього використовується чотири регістри довжиною 8 байтів з адресами 0, 2, 4, і 6 відповідно.

РЗП і **РПК** можна структурно об'єднати в 24-регістрову пам'ять **РП**, регістри 0...15 якої становлять **РЗП**, а інші 8 регістрів використовуються для збереження чотирьох восьмибайтных слів. Довжина регістру **РП** дорівнює 32 розрядам. Адреса регістру вказується в 5-розрядному регістрі адреси **АРП** (див. рис. 8). Операнд, що чи записується читається з **РП**, міститься в регістрі **РРП**. Читання і запис слова ініціюються відповідно сигналами **ЧтРП** і **ЗпРП**. Звертання до регістрів довжиною 8 байтів здійснюється за два цикли.

3.2. Інтерфейси основної і регістрової пам'яті.

Інтерфейс основної пам'яті (рис. 9) складається із сукупності шин **W, R, A, C,** $\mathbf{Z_{оп}}$. По шині запису **W** в основну пам'ять надходить інформація, яку необхідно записати в **ОП**. По шині читання **R** здійснюється передача інформації з **ОП** у процесор.

По шині адреси **A** на регістр адреси при звертанні у **ОП** надходить адреса слова, яке необхідно прочитати або записати в **ОП**. По шині управління **C** передаються сигнали **ЧтОП** і **ЗпОП**, що ініціюють операцію читання або запису. По шині ідентифікації з **ОП** надходить сигнал $\mathbf{Z_{оп}}$, що відзначає момент закінчення циклу звертання до пам'яті. Одиницею інформації, передаваємої по інтерфейсові основної пам'яті, є слово, що має розрядність, рівну ширині вибірки з **ОП**.

Можливий наступний спосіб підключення інформаційних і адресних шин. У схемі рис. 9 обмін інформацією між **ОП** і процесором відбувається через регістр **Z**, з'єднаний з магістраллю двома шинами. Попередньо сформована адреса слова зберігається на адресному регістрі **АР**, що зв'язаний з регістром **АОП** шиною **А.** Довжина **АР** дорівнює $\log_2 E$, де **Е** – ємність **ОП** у байтах, а довжина **АОП** визначається як $\log_2 E_C$, де E_C – ємність **ОП** у словах. Регістри **Z** та **РОП** рівні по довжині слову **ОП**.

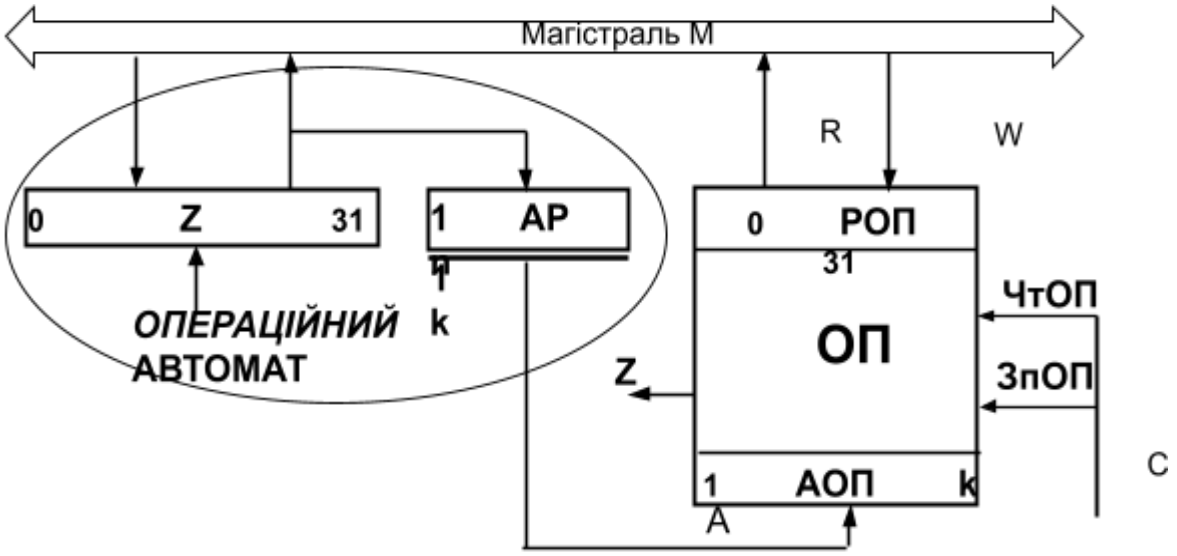

Рисунок 9 - Інтерфейс основної пам'яті

Обмін між процесором і **ОП** відбувається в такий спосіб. Нехай слово **А** необхідно записати в пам'ять за адресою **В**. У першому такті адреса **В** вибирається на регістр **АР**. В другому такті здійснюється завантаження операнда з запам'ятовуючої частини операційного автомата на регістр **Z**. Третій такт – такт безпосереднього запису – буде містити наступний набір мікрооперацій: **АОП:=АР(1:К), М:=Z, РОП:=М, ЗпОП**. Нульове значення сигналу $\mathbf{Z_{оп}}$ відзначить закінчення операції запису в **ОП**. Читання з **ОП** здійснюється відповідно до мікропрограми рис. 10.

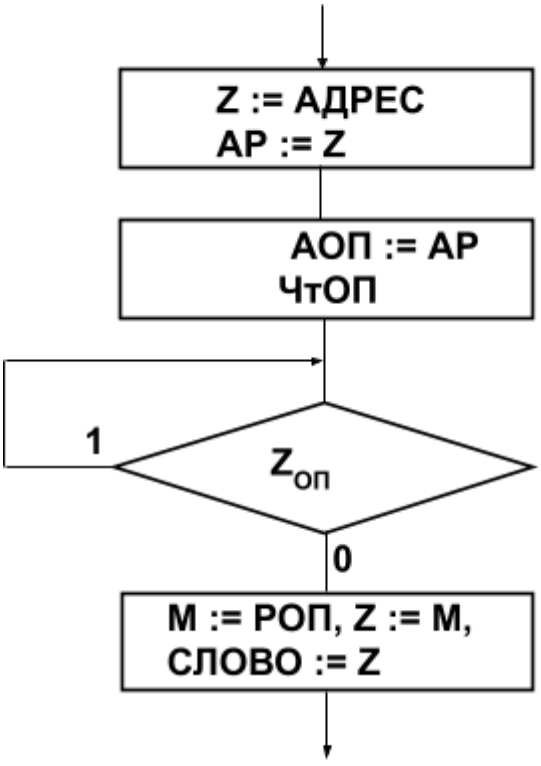

Рисунок 10 - Процедура звернення до основної пам'яті

Регістрова пам'ять є внутрішньою пам'яттю процесора. З'єднання операційного автомата процесора із **РП** може бути організоване за схемою, приведеній на рис. 11. Обмін між **РП** і процесором відбувається аналогічно обміну з **ОП**, але тому що час звертання до **РП** менше такту роботи операційного автомата, то в **РП** відсутня шина ідентифікації. Довжина регістру **РП** дорівнює 32 розрядам, довжина **АРП** – 5 розрядам. Тому що **РП** поєднує у собі **РЗП** і **РПК**, те необхідно організувати доступ до тих і інших регістрів.

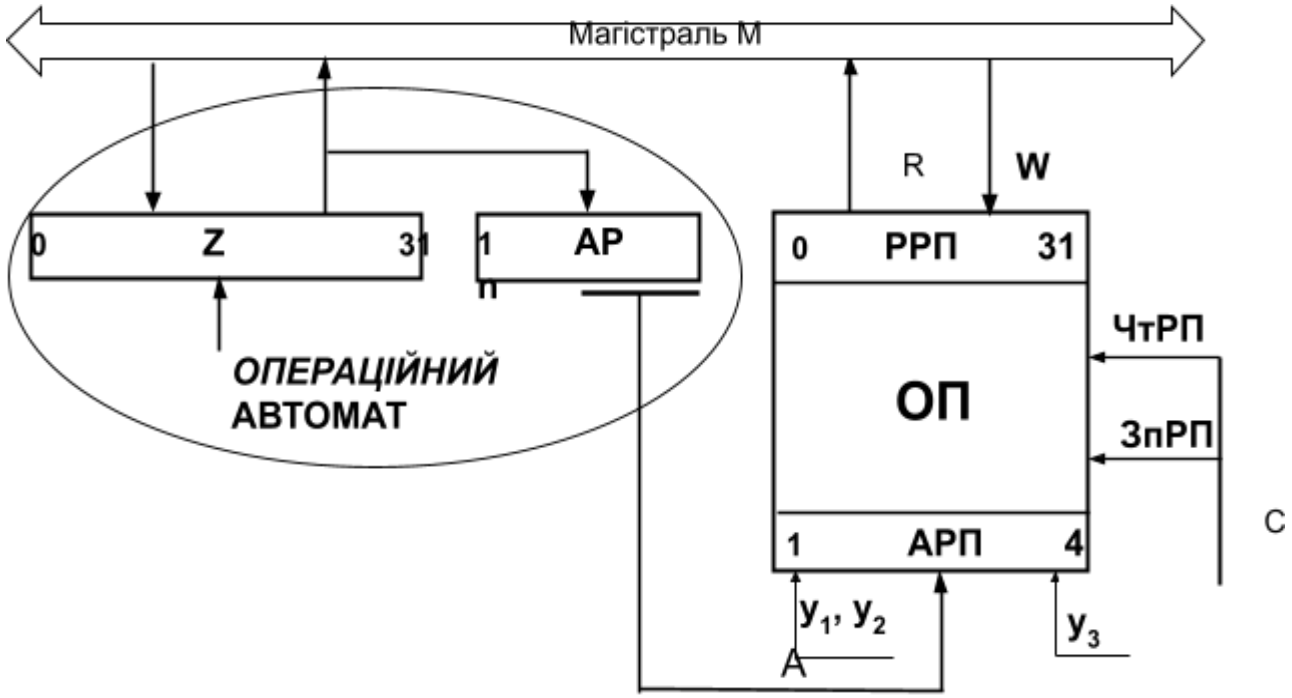

Рисунок 11 - Інтерфейс регістрової пам'яті

У форматі команди як під адресу **РЗП**, так і під адресу **РПК** приділяється 4-розрядне поле **R**; звертання до певного виду регістрів залежить від коду операції команди. Як зазначено раніше, перших 16 регістрів **РП** відведені під **РЗП**, а наступні 8 – під **РПК**. Тоді для звертання до **РЗП** необхідно установити розряд **АРП(0)** у нуль. Для цього вводиться мікрооперація $\mathbf{y_1}$**: АРП(0):=0**. Уведенням мікрооперації $\mathbf{y_2}$ **: АРП(0):=1** здійснюється доступ до **РПК**. Розряди **АРП(1:4)** обумовлюються полем **R** команди.

Для роботи з подвійними словами вводиться мікрооперація $\mathbf{y_3}$**: АРП(4):=1**. З її використанням можна змінити парну адресу **РП** на непарну, причому більшу на 1. Це необхідно в операціях із плавачою комою, коли довгий операнд зберігається на парі регістрів РПК.

3.3. Процедура вибірки команд

Програма, що становить собою послідовність команд різних форматів, розміщається в **ОП**. Для того, щоб виконати деяку команду, її необхідно вибрати з **ОП** на регістр команд **РК**. У залежності від формату обираної команди і ситуації, що передували вибірці даної команди, необхідно виконати різну сукупність мікрооперацій для передачі команди з **ОП** на **РК**.

Структура засобів, орієнтованих на реалізацію процедури вибірки команди з 32-розрядної **ОП**, представлена на рис. 12. Адреса обираної команди зберігається на лічильнику адреси команд **ЛАК**. **ЛАК** має довжину, рівну $\mathbf{\log_2 E - 1}$, тому що адреса команди будь-якого формату кратна півслову. Адресний регістр основної пам'яті **АОП** має довжину $\mathbf{\log_2 E_C}$. Значення останнього розряду **ЛАК** указує, із якого півслова, із першого або з другого, починається обирана команда.

Для позначки порядку проходження команд використовується тригер переходу **ТП.** Якщо команди виконуються в природному порядку, то **ТП:=0**. Після виконання команди переходу **ТП:=1**. Стан **ТП** обумовлює можливість використання інформації з раніше обраного слова, що зберігається на буферному регістрі. Якщо **ТП:=0**, то вміст буферного регістру **БР** може бути використовуватись для утворення наступної команди. Якщо **ТП:=1**, то управління передане

іншому слову, відсутньому на буферному регістрі, і, отже, вміст **БР** не може бути використане для формування наступної команди.

Слово **ОП** може містити цілком або команду тільки її частину. У 32-розрядному слові **ОП** може міститися тільки одна команда, наприклад, формату **RX**, або два поля по півслову, що належать різним командам. В останньому випадку використання буферного регістру для збереження другого півслова дозволяє виключити повторне читання з **ОП** того ж слова. Роль **БР** збільшується при 64-розрядному слові **ОП**.

Якщо обирана команда починається з першого півслова і довжина її дорівнює слову, то необхідно весь вміст регістру основної пам'яті **РОП** передати на регістр команди. Адреса наступної команди відрізняється від адреси обраної команди на два півслова, тому зміст **ЛАК** необхідно збільшити на 2. У випадку, коли обирана команда має формат **RR** , перше півслово, що становить собою команду, передається на **РК**, а розряди **РОП(16:31)** зберігаються на буферному регістрі. При цьому зміст лічильника адреси команди збільшується на одиницю.

Якщо обирана команда починається з другого півслова і є командою формату **RR** , то розряди **РОП(16:31)** передаються на регістр команди, зміст лічильника адреси команди збільшується на 1, буферний регістр не заповнюється.

Найбільшу складність становить випадок, коли команда починається з другого півслова і довжина її дорівнює слову. З **ОП** читається слово, що містить перше півслово обираної команди. Це півслово, тобто **РОП(16:31)** передається в регістр команди в розряди **РК(0:15)**, лічильник адреси збільшується на 2 і відбувається повторне звертання до **ОП**. Зі знову прочитаного слова перше півслово передається в останні розряди **РК(16:31)**, а друге півслово, що є новою командою або її частиною, запам'ятовується на буферному регістрі.

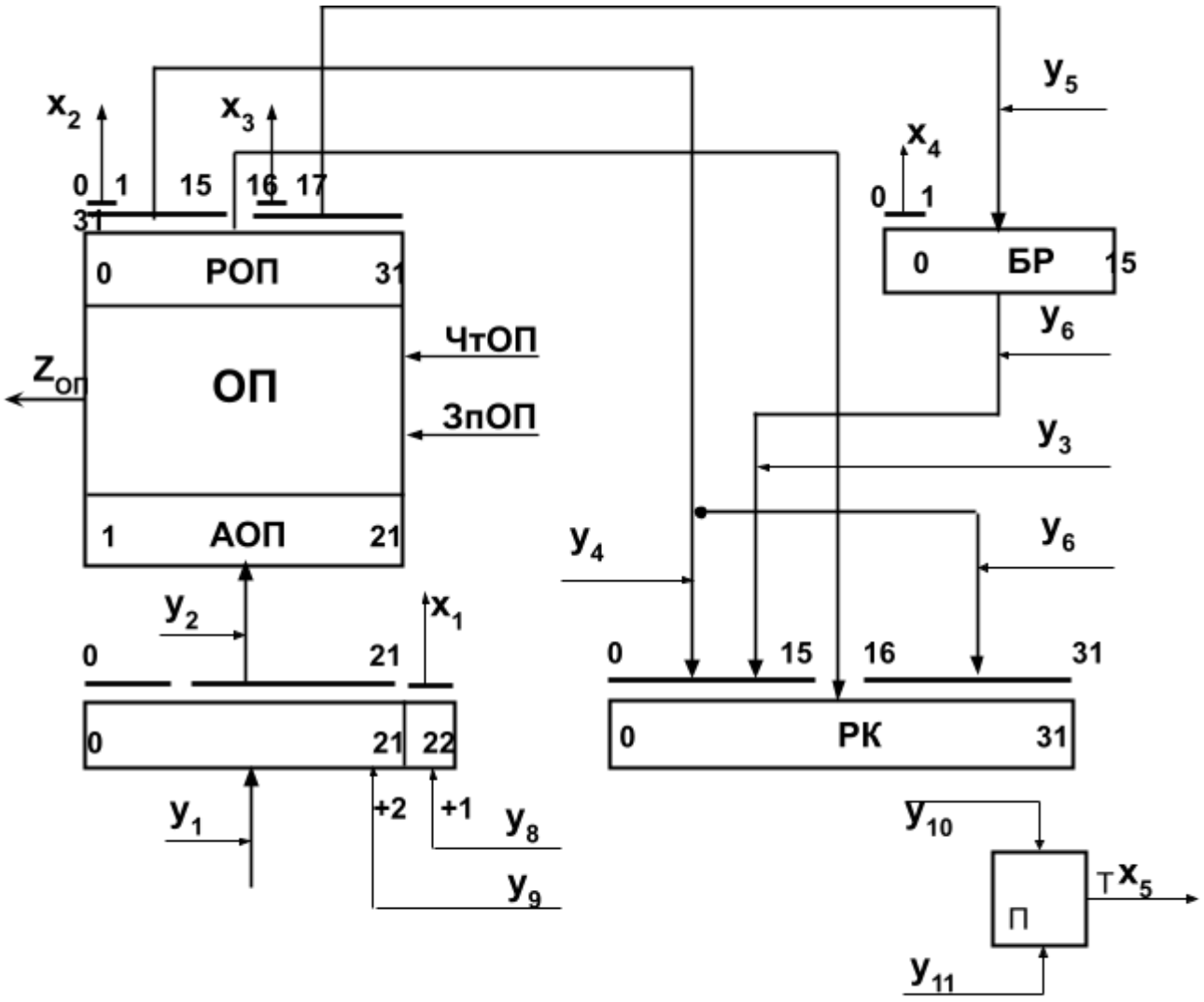

Рисунок 12 - Структура засобів, пов'язаних с вибіркою команди.

Послідовна вибірка команд і їхня обробка, при якій тригер переходу **ТП** зберігає нульове значення, має свої особливості. У цьому випадку вся команда, якщо вона має формат **RR** або частина команди формату **RX**, може зберігатися на буферному регістрі. Цей випадок має місце при одиничному значенні останнього розряду **СЧАК**. Значення перших двох розрядів **БР** указують на формат збереженої на ньому команди. У випадку команди **RR** достатньо вміст буферного регістру передати на **РК** і лічильник адреси збільшити на 1. Звертання до пам'яті не потрібно. Якщо ж на буферному регістрі знаходиться перше півслово команди формату **RX** , його необхідно передати на **РК(0:15)**, потім збільшити **ЛАК** на 2 і прочитати з **ОП** слово. Першим півсловом доповнюється регістр команд, а друге півслово заноситься на буферний регістр.

В автоматі з загальними мікрооперaціями для занесення інформації на регістр команд можна рекомендувати використовувати систему шин, представлену на рис. 13. Шина **Z** перед приєднанням її до регістру **РК** розділяється на дві частини **Z(0:15), Z(16:31)** . Кожне з цих полів може підключатися і до старших, і до молодших розрядів регістру **РК**.

Функціональна мікропрограма виб приведена на рис. 14.

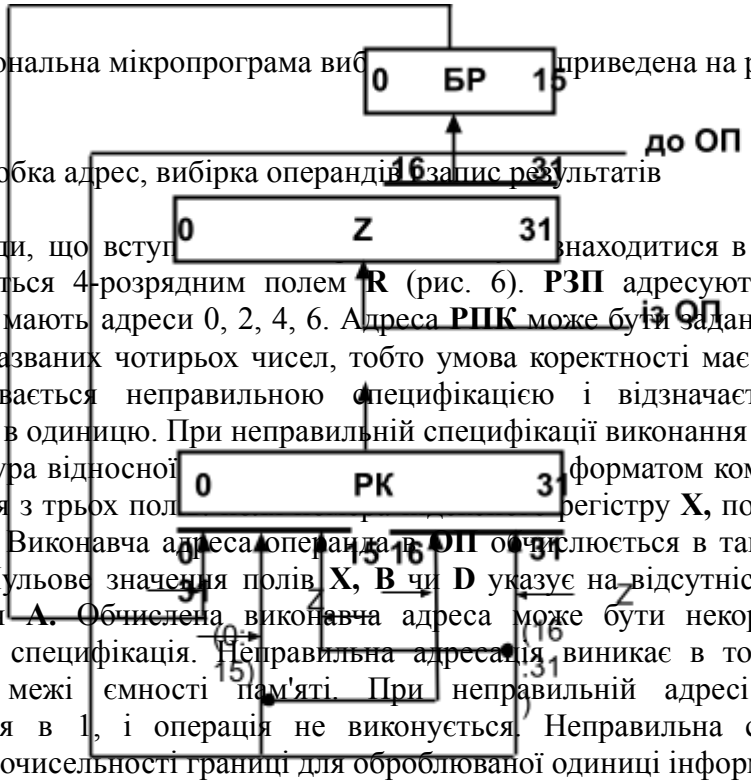

Рисунок 13 - Ланцюги, що використовуються при вибірці команди

3.4 Обробка адрес, вибірка операндів і запис результатів

Операнди, що вступ знаходитися в **РЗП**, **РПК** і в **ОП**. **РЗП** і **РПК** адресуються 4-розрядним полем **R** (рис. 6). **РЗП** адресуються числами від 0 до 15, регістри **РПК** мають адреси 0, 2, 4, 6. Адреса **РПК** може бути заданою некоректно, якщо вона відмінна від названих чотирьох чисел, тобто умова коректності має вигляд **R(0)∧R(3)=0**. Цей випадок називається неправильною специфікацією і відзначається установкою тригера переривання **S** в одиницю. При неправильній специфікації виконання операції припиняється.

Структура відносної форматом команди. Для команд **RX, RS** він складається з трьох пол регістру **X,** поля номера регістру бази **B**, поля зсуву **D**. Виконавча адреса операнда в **ОП** обчислюється в такий спосіб: **A:= РЗП[X] + РЗП[B] +D.** Нульове значення полів **X, B** чи **D** указує на відсутність відповідного доданка у вираженні для **A.** Обчислена виконавча адреса може бути некоректною, тобто порушена адресація або специфікація. Неправильна адресація виникає в тому випадку, якщо адреса виходить за межі ємності пам'яті. При неправильній адресі тригер переривання **A** встановлюється в 1, і операція не виконується. Неправильна специфікація є наслідком порушення цілочисельності границі для оброблюваної одиниці інформації.

Обробка адреси здійснюється операційним автоматом. Розглянемо процедуру одержання виконавчої адреси операнда на прикладі схеми рис.15. Значення адреси утворюється на накопичуваючому додавачі довжиною 24 розряду. З регістру команди **РК** на додавач передається зміщення **D**. Потім аналізується поле індексу **X.** Якщо воно не дорівнює нулю, то на адресний регістр **РП** у розряди 1...4 передаємо вміст поля **X.** При цьому виконуємо мікрооперацію **АРП(0):=0**, тому що звертання відбувається до **РЗП**. Після подачі сигналу **ЧтРП** одержуємо значення індексу на регістрі **РРП**. На додавачі значення індексу складається зі зміщенням. Аналогічно утворюється значення базової адреси, на яке збільшується значення суми.

Правильність адресації і специфікації обчисленої адреси перевіряється в такий спосіб. Обчислена адреса операнда може містити 24 розряди. При правильній адресації старші розряди

адреси, що виходить за межі ємності **ОП**, повинні бути нульовими. Перевірка на специфікацію різна для різних команд і залежить від довжини оброблюваних слів. Наприклад, якщо команда оперує зі словами, то два останніх розряди адреси повинні бути нульовими.

Обмін з **ОП** відбувається тільки словами **ОП**. Якщо операндами виконуваної команди є частини слова, то необхідно спочатку прочитати цілком слово на регістр **Z,** потім на регістр пам'яті операційного автомата передати необхідне півслово або байт. При роботі з півсловами старшим розрядам слова присвоюється значення знакового розряду.

Функціональна мікропрограма процедури вибірки операнда з **ОП** зображена на рис. 16.

При запису в **ОП** операндів довжиною в півслово або байт, слово, що буде містити цей операнд, читається на регістр **Z** потім у цьому слові відповідна його частина заміняється операндом. Знову сформоване слово записується в **ОП** по тій же адресі. Наприклад, нехай необхідно байт, що зберігається в молодших розрядах регістру **A1** записати в **ОП** за адресою **XXX…X10**. Для цього виконується послідовність дій, обумовлених мікропрограмою рис.17.

Якщо довжина операнду перевищує довжину слова пам'яті, то читання і запис операндів здійснюється за кілька звертань до **ОП**

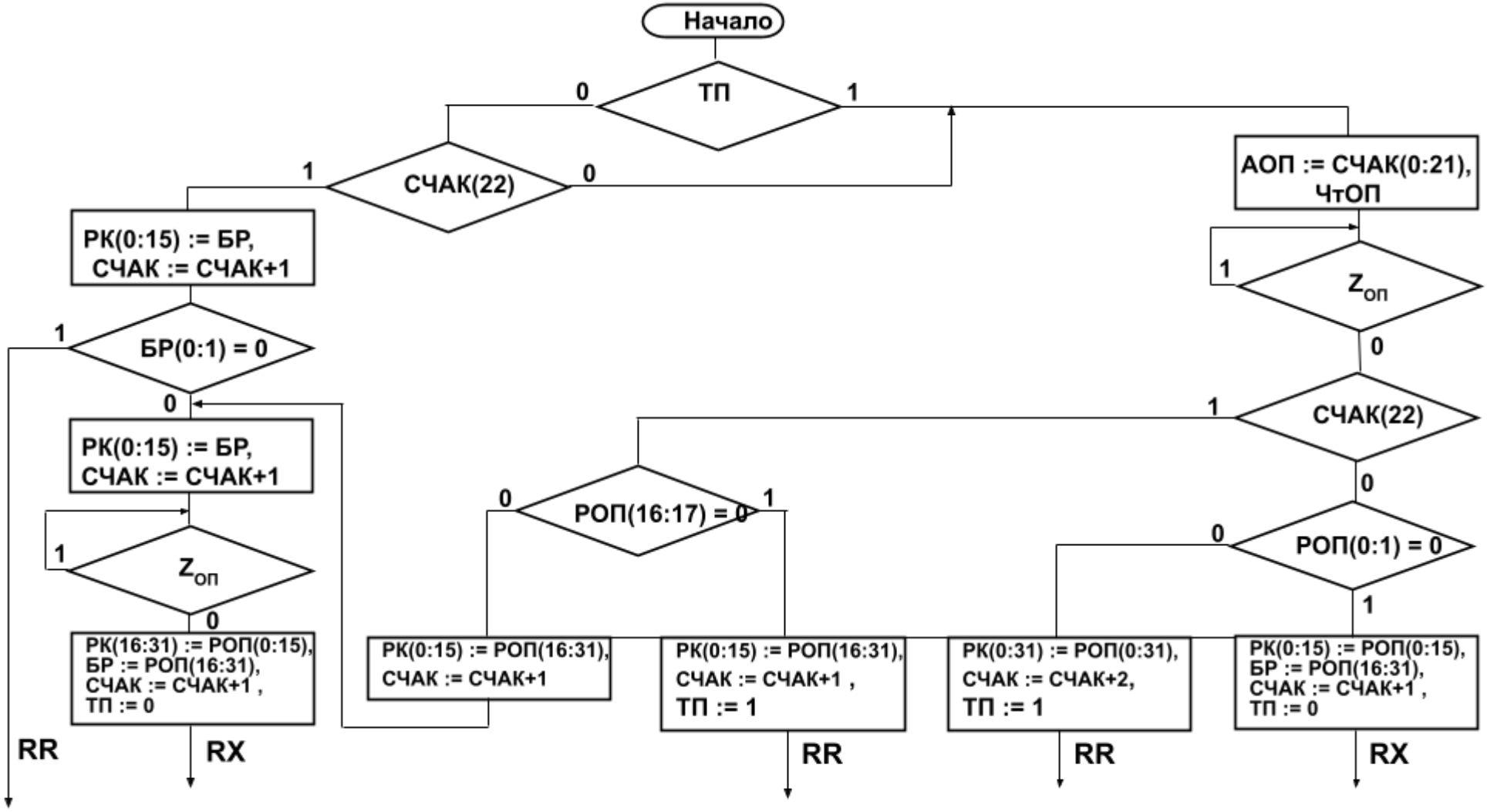

Рисунок 14 - Процедура вибірки команд формату **RR** та **RX**

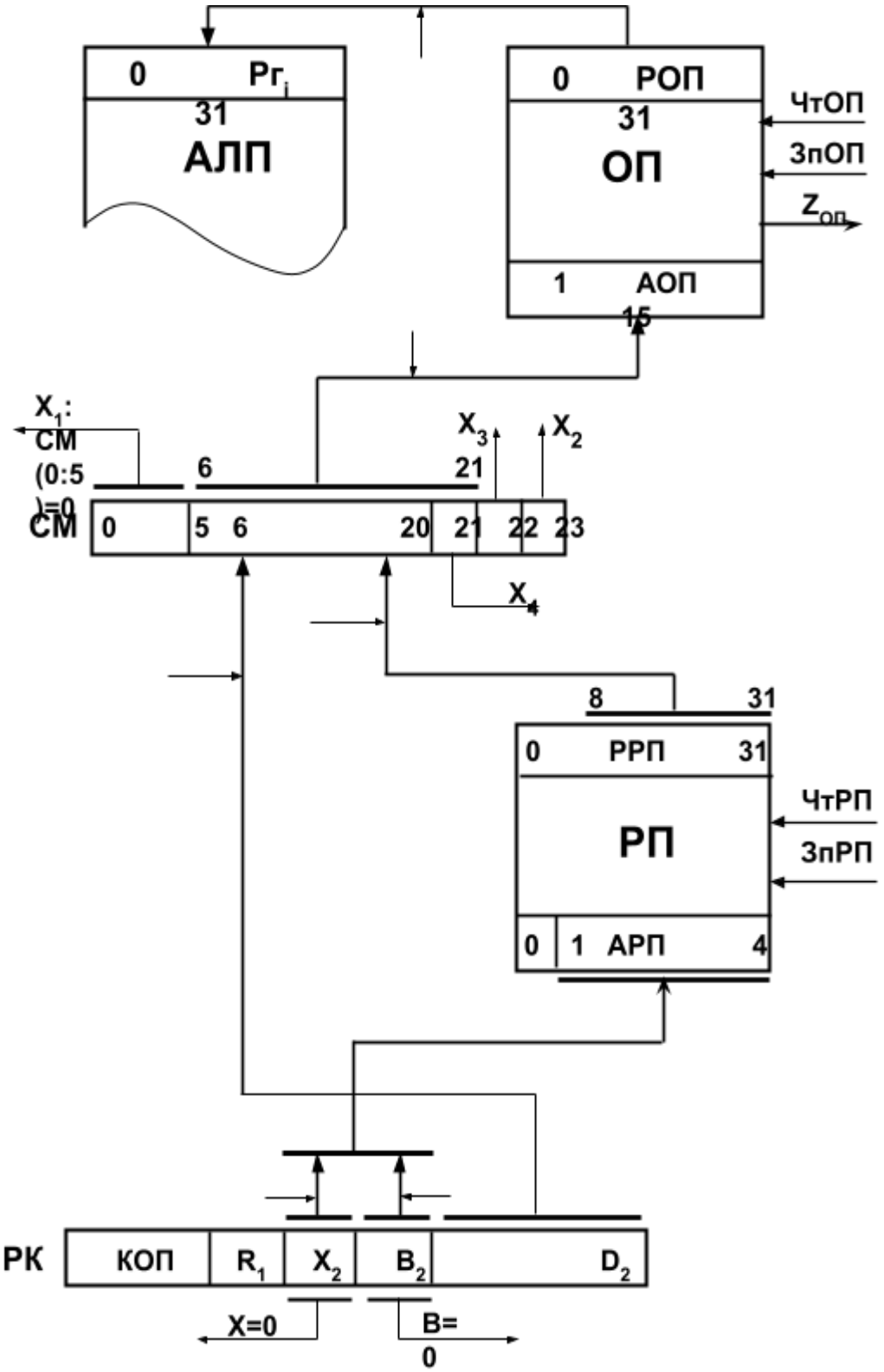

Рисунок 15 - Схема формування виконавчої адреси

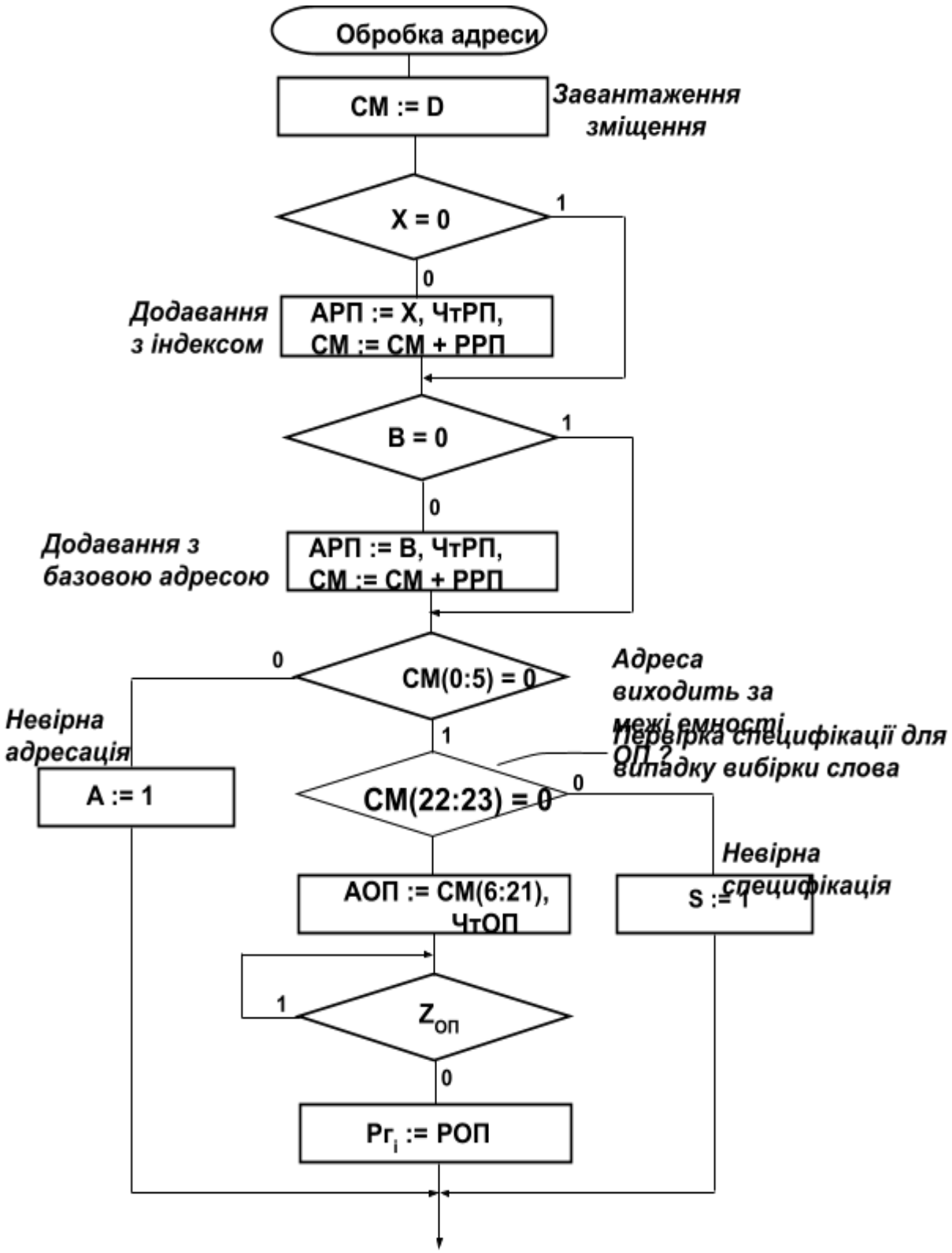

Рисунок 16 - Процедура обробки відносної адреси та вибірки операнду з оперативної пам'яті

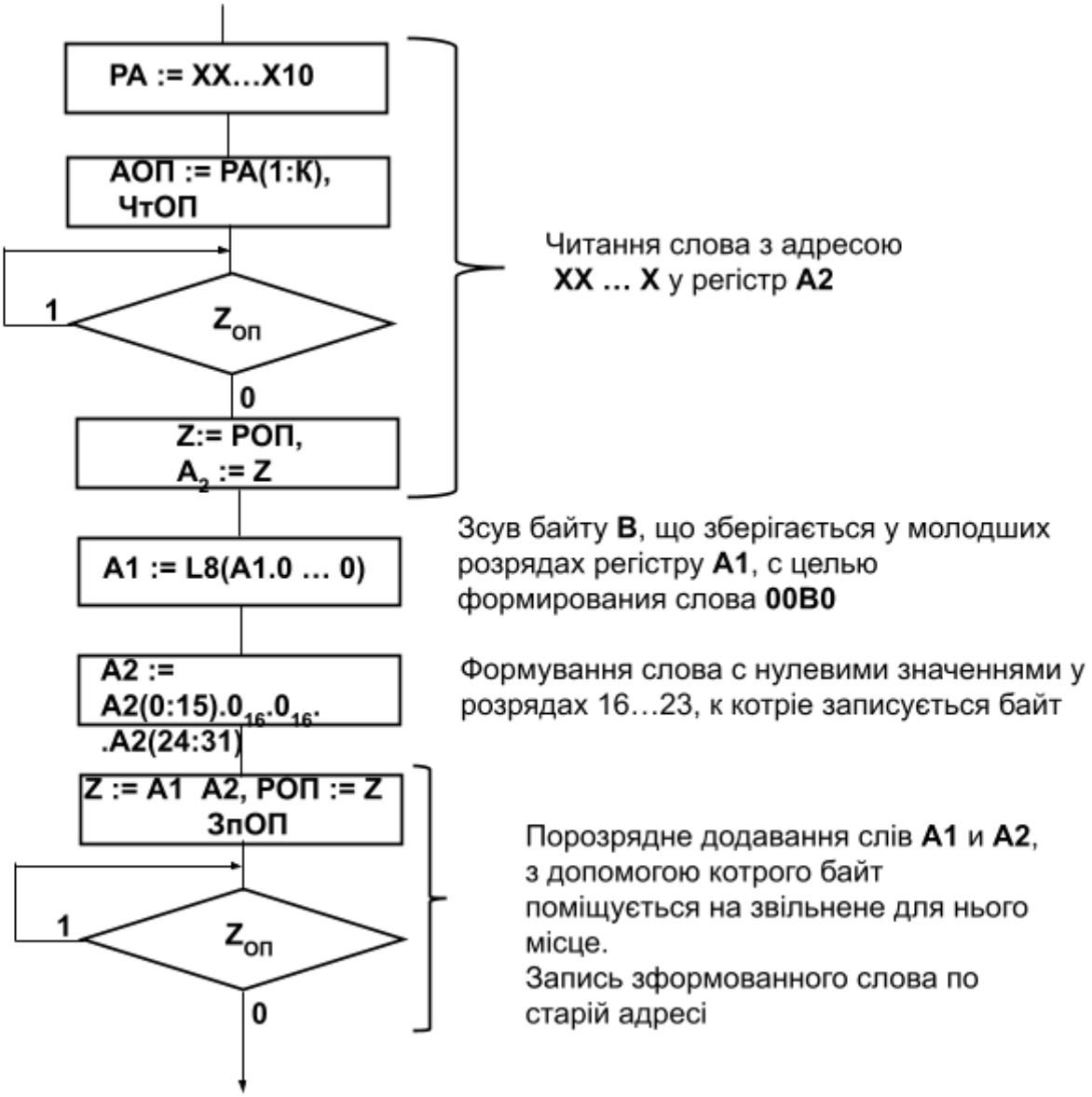

Рисунок 17 - Принцип побудови процедури запису байта з адресою **XX …X10**

4 ТЕМАТИКА І ЗМІСТ КУРСОВОГО ПРОЕКТУ

4.1 Тематика курсового проектування

Об'єктом проектування є процесор, призначений для реалізації заданого набору команд із множини команд обчислювальних машин IBM/370. Вихідні дані для проектування обумовлюються ***номером завдання***, ***котрий визначається викладачем*** і вибираються з таблиць 4 та 5 і 6..

4.2 Порядок виконання курсового проекту

Курсове проектування розділяється на ряд етапів, що включають у себе:

- розробку функціональних мікропрограм виконання заданих команд;
- об'єднання функціональних мікропрограм окремих операцій та побудова об'єднаної функціональної мікропрограми;
- розробку структурної схеми операційного автомату процесора;
- синтез керуючого автомата із програмувальною логікою.

4.2.1 Розробка функціональних мікропрограм виконання команд

Розробка функціональних мікропрограм (**МП**) здійснюється на основі функціонального опису команд [4]. Для кожної операції будується функціональна **МП**, що відображає всі етапи її виконання: вибірку команди; формування адреси і вибірку операндів, обробку операндів відповідно до обраного алгоритму виконання операції; запис результату; установку ознаки результату; дії, зв'язані з обробкою переривань. При розробці алгоритмів виконання арифметичних операцій можна користуватися [5].

Функціональні **МП** будуються або у вигляді змістовних граф-схем алгоритмів (**ГСА**), або у вигляді текстів мікропрограм з використанням **ФМ**-языка і встановлених умовних позначок [3]. При їхній побудові враховуються стани процесора: **СТІП - РОБОТА; ЗАДАЧА - СУПЕВІЗОР; МАСКА - ПЕРЕРИВАННЯ, ВИКОНАННЯ ПРОГРАМИ – ЧЕКАННЯ**, а також особливі випадки, що приводять до переривань програми.

4.2.2 Розробка об'єднаної функціональної мікропрограми

Побудова об'єднаної функціональний **МП** здійснюється з урахуванням евристичних принципів об'єднання **МП**. При побудові об'єднаної **МП**, що відображає функціонування процесора, який реалізує заданий набір операцій, варто виходити з умови мінімізації числа операторних і умовних вершин у **ГСА**. Склеювання подібних між собою ділянок **МП** окремих операцій варто здійснювати для всіх етапів виконання команди.

На основі об'єднаної **МП** здійснюється виділення списків мікрооперацій і логічних умов, необхідних і достатніх для реалізації заданого набору операцій у процесорі, а також списків вхідних, вихідних і внутрішніх слів мікропрограми.

При розробці структурно-функціональної **МП** (**СФМП**) кожної мікрооперації з функціональної мікропрограми ставляться у відповідність множина керуючих сигналів, що ініціюють їх виконання в рамках розробленої структури. При переході від функціональної **МП** до структурно-функціональної враховуються всі особливості структури процесора, до якої відносяться довжина слова **ОП**, розрядність схем комбінаційної частини **ОА**, розподіл слів по регістрах пам'яті **ОА** і т.д. Таким чином, структурно-функціональна **МП** повинна описувати алгоритм функціонування процесора в термінах керуючих і інформаційних сигналів. **СФМП** утворюється шляхом представлення кожної операторної й умовної вершини графа функціональної **МП** сукупностями мікрооперації і логічних умов, необхідних для інтерпретації функціональних операторів.

При побудові **СФМП** необхідно враховувати можливість суміщення окремих мікрооперацій із функціональної **МП** у рамках проектуємої структури. Мікрооперації, що є сумісними у функціональної **МП**, можуть стати несумісними в **СФМП** і навпаки.

4.2.3 Розробка структурної схеми операційного автомата процесора

Розроблюєвама структурна схема повинна орієнтуватися на виконання чотирьох зазначених у завданні команд. Побудова структурної схеми операційного автомата здійснюється з урахуванням списків мікрооперацій, логічних умов і списку вхідних, вихідних і внутрішніх слів, на основі яких вибирається мінімальний набір операційних елементів, схем формування інформаційних сигналів і склад регістрів у пам'яті операційного автомата. На структурній схемі **ОА** відображається його зв'язок з регістровою й оперативною пам'яттю, що може бути реалізовано на основі індивідуальних шин або загальної магістралі.

При виборі структурних рішень, зв'язаних з різними витратами устаткування і швидкодією, варто самостійно приймати й обґрунтовувати оптимальні рішення. Структурна схема розробляється і викреслюється з точністю до окремого операційного елементу. На схемі указуються всі керуючі й інформаційні сигнали. Нетривіальні рішення обґрунтовуються в пояснювальній записці.

4.2.4 Синтез керуючого автомата із програмувальною логікою

Для синтезу керуючого автомату (**КА)** необхідно вибрати найбільш розгалужену ділянку **МП**, що має один вхід, єдиний вихід і число операторних і умовних вершин 50...64. Для цієї ділянки будується кодована ГСА. Синтез **КА** здійснюється методами, викладеними в [2]. Необхідно розробити структурну й електричну принципову схеми **КА**. Розробка принципової схеми здійснюється на основі заданої системи елементів.

Побудова **КА** з програмувальною логікою містить у собі вибір формату мікрокоманди з урахуванням оптимального розподілу мікрооперацій по її полям і заданому способі адресації. Необхідно прагнути мінімізувати витрати пам'яті на збереження мікрокоманд і витрати устаткування на інтерпретацію мікрокоманд. Для **КА** будується структурна схема, що включає до себе схеми формування адрес мікрокоманд і схеми формування керуючих сигналів. Для обраного формату мікрокоманди здійснюється кодування керуючих і інформаційних сигналів і складається таблиця прошивання **ПЗП** мікрокоманд для заданого фрагмента об'єднаної **МП**.

4.3. Зміст курсового проекту

Кожен етап виконання курсового проекту повинен знайти відображення в пояснювальній записці, у якій зводяться вихідні і розрахунково-пояснювальні матеріали, зв'язані з розробкою проекту. Основні графічні матеріали оформляються у вигляді креслень відповідно до ЄСКД.

4.3.1.Склад пояснювальної записки

1. Анотація.
2. Завдання.
3. Вступ.
4. Опис функціональної організації спроектованого процесора. Для кожної операції, реалізованої процесором, приводиться опис формату команди і форматів даних, словесний опис алгоритму виконання команди, особливих випадків, що приводять до переривання, принципів установки ознаки результату.
5. Функціональні мікропрограми виконання окремих команд і їхній опис. Однакові частини мікропрограм різних команд можна не дублювати, заміняючи одним блоком, таким як «вибірка команди формату **RR**», «вибірка команд форматів **RX, RS**», «обчислення адреси другого операнду для команди формату **RX**» і т.п. Об'єднана мікропрограма оформляється у вигляді окремого документа (креслення).
6. Опис процедури синтезу структурної схеми операційного автомата і його зв'язків з регістровою й основною пам'яттю. Приводиться перелік усіх слів, використаних при написанні **МП,** їхніх типів, списки мікрооперацій і логічних умов. Усі нетривіальні рішення обґрунтовуються. Структурна схема оформляється у вигляді окремого документу (креслення).
7. Синтез керуючого автомата із програмувальною логікою. Приводиться опис структурної схеми **КА** відповідно до заданого типу адресації. Здійснюється обґрунтування

вибору форматів мікрокоманд. На кодованій **ГСА** відзначаються номери мікрокоманд, що реалізують відповідні операторні й умовні вершини. Будується карта прошивання **ПЗП** мікрокоманд із коментарями. Кодована **ГСА** виконується у вигляді окремого документа (креслення).

8. Висновок.
9. Перелік посилань.

4.3.2 Склад графічної частини

1. Процесор. Схема електрична структурна.
2. Процесор. Мікропрограма функціональна об'єднана.
3. Керуючий автомат. ГСА кодована.
4. Керуючий автомат. Схема електрична структурна.
5. Керуючий автомат. Схема електрична принципова.

Таблица 4 - Перелік шістнадцеричных кодов команд, выполняемых процессором

	АМ-ХХ1	**АМ-ХХ2**	**АМ-ХХ3**	**АМ-ХХ4**	**АМ-ХХ5**
1.	1A 6D 45 D4	4A 20 46 FA	5E 21 47 FC	54 31 05 DD	5A 2B 45 D7
2.	5A 3D 07 94	1E 6E 87 F9	14 7F 8A F0	1A 24 86 D4	4A 6B 07 FA
3.	4A 24 05 91	5E 30 45 FD	54 3F 46 FB	5A 3A 47 94	1E 6B 8A F9
4.	1E 6A 86 D7	14 7C 07 DC	1A 6D 87 F8	4A 6C 06 91	5E 33 86 FD
5.	5E 3A 47 FA	54 3E 05 FC	5A 3D 45 DD	1E 3C 46 D7	14 6D 47 DC
6.	14 7A 06 F9	1A 7E 86 F0	4A 3F 07 D4	5E 2E 87 FA	54 3D 45 FC
7.	54 2E 46 FD	5A 28 47 FB	1E 78 05 94	14 7A 45 F9	1A 3F 46 F0
8.	1A 6E 87 DC	4A 69 06 F8	5E 24 86 91	54 7C 07 FD	5A 21 87 FB
9.	5A 3E 8F FC	1E 29 8E DD	14 7A 88 D7	1A 6E 89 DC	4A 7F 06 F8
10.	4A 7E 07 F0	5E 22 87 D4	54 6D 06 FA	5A 30 86 FC	1E 6B 07 DD
11.	1E 69 8D FB	14 2D 8C 94	1A 6A 46 F9	4A 20 47 F0	5E 2D 8E DD
12.	5E 29 89 F8	54 39 07 91	5A 3A 87 FD	1E 7E 06 FB	14 60 05 D4
13.	14 2D 8B DD	1A 60 05 D7	4A 3D 45 DC	5E 3E 46 F8	54 39 86 94
14.	54 39 8C D4	5A 2B 86 FA	1E 2E 88 FC	14 69 87 DD	1A 2B 47 91
15.	1A 3B 8E 94	4A 32 47 F9	5E 6E 05 F0	54 28 45 D4	5A 32 46 D7
16.	5A 2F 87 91	1E 7D 06 FD	14 7E 86 FB	1A 29 8E 94	4A 7D 06 FA
17.	4A 2B 45 D7	5E 23 46 DC	54 7E 47 F8	5A 22 05 91	1E 23 87 F9
18.	1E 6B 88 FA	14 24 8B FC	1A 69 06 DD	4A 2D 86 D7	5E 24 45 FD
19.	5E 6D 05 F9	54 38 45 F0	5A 29 46 D4	1E 39 47 FA	14 79 8C DC
20.	14 3D 8A FD	1A 79 07 FB	4A 2D 87 94	5E 60 06 F9	54 38 05 FC
21.	54 3F 47 DC	5A 3B 05 F8	1E 39 45 91	14 2B 46 FD	1A 3B 86 F0
22.	1A 6F 06 FC	4A 34 86 DD	***5E 3B 07*** D7	54 32 87 DC	5A 34 47 FB
23.	5A 2C 46 F0	1E 2F 8D D4	14 2F 8D FA	1A 7D 45 FC	4A 2F 8F F8
24.	4A 31 87 FB	5E 2B 06 94	54 2B 86 F9	5A 23 8C F0	1E 6B 46 DD
25.	1E 24 8F F8	14 6B 46 91	1A 6B 47 FD	4A 24 05 FB	5E 2B 87 D4
26.	5E 3A 07 DD	54 33 87 D7	5A 6D 06 DC	1E 79 86 F8	14 6B 45 94
27.	14 6C 05 D4	1A 6B 45 FA	4A 3D 46 FC	5E 38 47 DD	54 33 07 91
28.	54 3C 86 94	5A 6D 07 F9	1E 3F 87 F0	14 3B 8F D4	1A 6D 8A D7
29.	1A 7A 47 91	4A 3D 05 FD	5E 2C 45 FB	54 34 46 94	5A 3D 86 FA
30.	5A 2E 06 D7	1E 6B 89 DC	14 6F 07 F8	1A 2F 87 91	4A 3F 47 F9

Примітка: 1. Довжина слова оперативної пам’яті для усіх варіантів - 8 байт.

2. Обсяг оперативної пам’яті та тип керуючого автомата визначаються за таблицями 5 и 6.

Таблиця 5 - Обсяг пам’яті та тип керуючого автомата

	АМ-ХХ1		**АМ-ХХ2**		**АМ-ХХ3**		**АМ-ХХ4**		**АМ-ХХ5**	
	$V_{оп}$	Тип УА	$V_{оп}$	Тип УА	$V_{оп}$	Тип УА	$V_{оп}$	Тип УА	$V_{оп}$	Тип УА
1	256	1	512	3	1024	2	2048	1	4096	3
2	512	2	1024	1	2048	3	4096	2	8192	1
3	1024	3	2048	2	4096	1	8192	3	256	2
4	2048	1	4096	3	8192	2	256	1	512	3
5	4096	2	8192	1	256	3	512	2	1024	1
6	8192	3	256	2	512	1	1024	3	2048	2
7	256	1	512	3	1024	2	2048	1	4096	3
8	512	2	1024	1	2048	3	4096	2	8192	1
9	1024	3	2048	2	4096	1	8192	3	256	2
10	2048	1	4096	3	8192	2	256	1	512	3
11	4096	2	8192	1	256	3	512	2	1024	1
12	8192	3	256	2	512	1	1024	3	2048	2
13	256	1	512	3	1024	2	2048	1	4096	3
14	512	2	1024	1	2048	3	4096	2	8192	1
15	1024	3	2048	2	4096	1	8192	3	256	2
16	2048	1	4096	3	8192	2	256	1	512	3
17	4096	2	8192	1	256	3	512	2	1024	1
18	8192	3	256	2	512	1	1024	3	2048	2
19	256	1	512	3	1024	2	2048	1	4096	3
20	512	2	1024	1	2048	3	4096	2	8192	1
21	1024	3	2048	2	4096	1	8192	3	256	2
22	2048	1	4096	3	8192	2	256	1	512	3
23	4096	2	8192	1	256	3	512	2	1024	1
24	8192	3	256	2	512	1	1024	3	2048	2
25	256	1	512	3	1024	2	2048	1	4096	3
26	512	2	1024	1	2048	3	4096	2	8192	1
27	1024	3	2048	2	4096	1	8192	3	256	2
28	2048	1	4096	3	8192	2	256	1	512	3
29	4096	2	8192	1	256	3	512	2	1024	1
30	8192	3	256	2	512	1	1024	3	2048	2

Таблица 6 - Тип управляющего автомата

1.	Природна адресація з перевіркою однієї логічної умови та одним полем повної адреси в форматі мікрокоманди
2.	Примусова адресація з перевіркою двох логічних умов та одним полем повної адреси в форматі мікрокоманди
3.	Примусова адресація з перевіркою однієї логічної умови та одним полем повної адреси в форматі мікрокоманди

ПЕРЕЛІК ПОСИЛАНЬ

1. Каган Б.М. Электронные вычислительные машины и системы: Учеб. пособие для вузов. – М.: Энергоатомиздат, 1991. –592 с.
2. Майоров С.А., Новиков Г.И. Структура электронных вычислительных машин. - Л: Машиностроение, 1989.
3. Полин Е.Л. Язык функционального микропрограммирования (ФМ-язык): Учебное пособие. – К: ИСИО, 1995.
4. Принципы работы системы IBM/370. – М: Мир, 1975.
5. Дроздов Е.А., Комарницкий В.А., Пятибратов А.П. Электронные вычислительные машины единой системы. – М.: Машиностроение, 1981.