МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІГІВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

МОДЕЛЬ ЕОМ З АКУМУЛЯТОРНОЮ АРХІТЕКТУРОЮ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
“Архітектура комп’ютерів”
для студентів спеціальності
123 – “Комп’ютерна інженерія”

Затверджено
на засіданні кафедри
інформаційних і комп'ютерних систем
Протокол № 10
від 30 березня 2017 р.

Чернігів ЧНТУ 2017

Модель ЕОМ з акумуляторною архітектурою. Методичні вказівки до лабораторних робіт з дисципліни “Архітектура комп`ютерів” для студентів спеціальності 123 – "Комп'ютерна інженерія". / Укл. Роговенко А.І., Красножон О.В., Красножон А.В. – Чернігів: ЧНТУ, 2017.– 55 с. Укр. мовою.

Укладачі: РОГОВЕНКО АНДРІЙ ІВАНОВИЧ, старший викладач кафедри інформаційних та комп'ютерних систем
КРАСНОЖОН ОЛЕКСІЙ ВАСИЛЬОВИЧ, старший викладач кафедри інформаційних та комп'ютерних систем
КРАСНОЖОН АНДРІЙ ВАСИЛЬОВИЧ, кандидат техничних наук, доцент кафедри електричних систем і мереж

Відповідальний за випуск: ЗАЙЦЕВ СЕРГІЙ ВАСИЛЬОВИЧ, завідувач кафедри інформаційних і комп'ютерних систем, доктор технічних наук, доцент

Рецензент: НЕСТЕРЕНКО СЕРГІЙ ОЛЕКСАНДРОВИЧ, кандидат технічних наук, доцент кафедри інформаційних і комп'ютерних систем Чернігівського національного технологічного університету.

ЗМІСТ

ВСТУП

Методичні вказівки призначені для вивчення взаємодії пристроїв в структурі ЕОМ за допомогою програмної моделі деякої абстрактної навчальної ЕОМ, яка програмується на мові асемблера.

Шлях сучасного програміста починається зі знайомства з мовою (мовами) високого рівня і все його спілкування з комп'ютером проходить з використанням таких мов.

У багатьох випадках знання операторів мови високого рівня, структури даних і способів їх обробки є достатнім для створення різних прикладних програм. Однак по-справжньому вирішувати проблеми, пов'язані з управлінням різною, особливо нестандартною, апаратурою (спеціалізовані комп'ютерні системи) неможливо без знання асемблера. Не випадково практично всі компілятори мов високого рівня містять засоби зв'язку своїх модулів з модулями на асемблері або підтримують вихід на асемблерний рівень програмування.

Однак проводити початкове навчання програмуванню на низькому рівні з розглядом механізмів взаємодії пристроїв на реальну мову, наприклад x86 на персональній ЕОМ, не завжди зручно. У цьому випадку між користувачем і апаратурою ЕОМ присутній операційна система (ОС) яка суттєво обмежує бажання користувача експерементувати з апаратними засобами. Для подолання цих обмежень необхідно мати глибокі знання як ОС, так і апаратних засобів ЕОМ.

Пропонована для використання програмна модель навчальної ЕОМ відображає всі основні особливості систем команд і структур сучасних простих ЕОМ, включає в себе, крім процесора і пам'яті, моделі декількох типових зовнішніх пристроїв. Модель дозволяє вивчити основи пропрограмування на низькому рівні, питання взаємодії різних рівнів пам'яті в складі ЕОМ і способи взаємодії процесора з зовнішніми пристроями.

1 СТРУКТУРА ЕОМ

Модель ЕОМ включає процесор, оперативну пам'ять (ОП) і надоперативну пам'ять (СОП), пристрій введення (ПВВ) і пристрій виведення (ПВИВ). Процесор в свою чергу складається з центрального пристрою управління (ПУ) арифметичного пристрою (АП) і системних регістрів (CR, PC, SP і ін.). Структурна схема ЕОМ показана на малюнку 1.1.

В комірках ОЗП зберігаються команди і дані. Ємність ОЗП становить 1000 комірок. За сигналом MWr виконується запис вмісту регістра данних (MDR) в комірку пам'яті з адресою, зазначеною в регістрі адреси (MAR). За сигналом MRd відбувається зчитування - вміст комірки пам'яті з адресою, що містяться в MAR, передається в MDR.

Надоперативна пам'ять з прямою адресацією містить десять регістрів загального призначення R0-R9. Доступ до них здійснюється (аналогічно доступу до ОЗП) через регістри RAR і RDR.

АП здійснює виконання однієї з арифметичних операцій, визначається кодом операції (COP), над вмістом акумулятора (Acc) і регістра операнда (DR). Результат операції завжди поміщається в Acc. При завершенні виконання операції АУ виробляє сигнали ознак результату: Z (дорівнює 1, якщо результат дорівнює нулю); S (дорівнює 1, якщо результат негативний); OV (дорівнює 1, якщо при виконанні операції сталося переповнення розрядної сітки). У випадках, коли ці умови не виконуються, відповідні сигнали мають нульове значення.

У моделі ЕОМ передбачено зовнішні пристрої двох типів. По-перше, це регістри IR і OR, які можуть обмінюватися з акумулятором за допомогою безадресних команд in (Acc: = IR) і out (OR: = Ac). По-друге, це набір моделей зовнішніх пристроїв, які можуть підключатися до системи і взаємодіяти з нею відповідно до закладених в моделі алгоритмів. Кожній зовнішній пристрій має ряд програмно-доступних регістрів, може мати власний оглядач (вікно видимих елементів). Детальніше ці зовнішні пристрої описані в розділі 6.

ПУ здійснює вибірку команд з ОП в послідовності, визначається неушкодженим природним порядком виконання команд (в порядкузбільшення адресів команд в ОП) або командами передачі управління; вибірку з ОП операндів, що задаються адресами команди; ініціювання виконання операції, запропонованої командою; зупинка чи перехід до виконання наступної команди.

В сверхоперативной пам'яті в модель включені регістри загального призначення (РЗП), і може підключатися модель кеш-пам'яті.

До складуОЗП ЕОМ входять:

1. PC - лічильник адресу команди, що містить адресу поточної команди;
2. CR - регістр команди, що містить код команди;
3. RB - регістр базової адреси, що містить базову адресу;
4. SP - покажчик стека, що містить адреса верхівки стека;
5. RA - регістр адреси, що містить виконавчий адресу при непрямій

адресації.

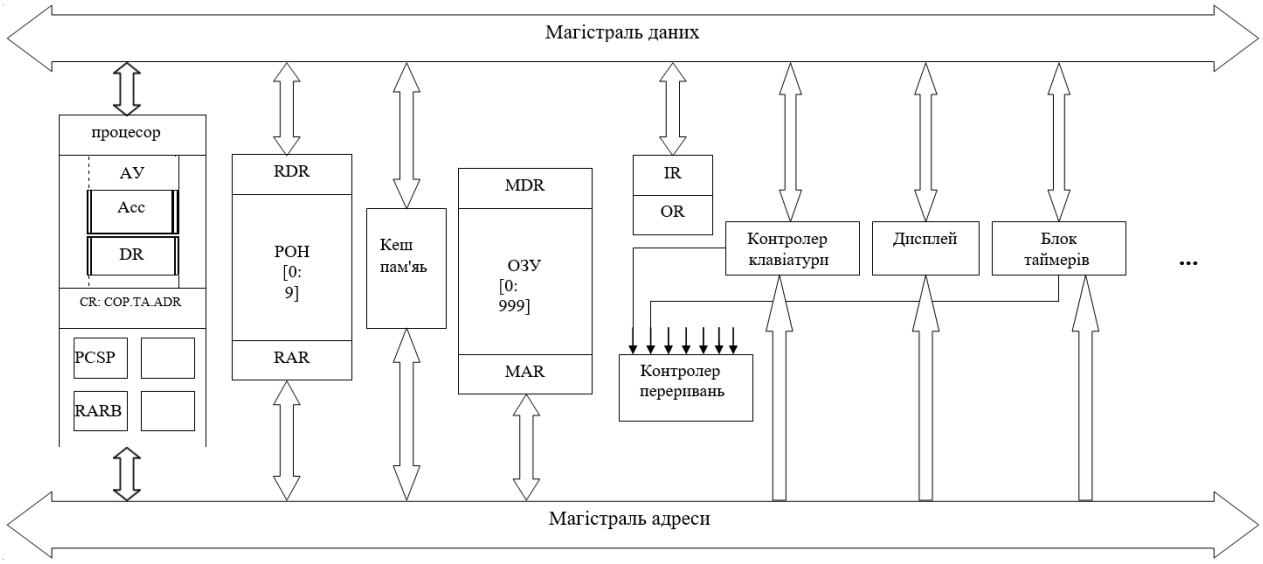

Рисунок 1.1 – Загальна структура навчальної ЕОМ

Регістри Acc, DR, IR, OR, CR і всі комірки ОЗП і РОН мають довжину 6 десяткових розрядів, регістри PC, SP, RA і RB - 3 розряду.

1.1 Подання даних в моделі

Дані в ЕОМ представляються у форматі, показаному на рисунку1.2. Це цілі десяткові числа, що змінюються в діапазоні "-99 999 ... +99 999", що містять знак і 5 десяткових цифр.

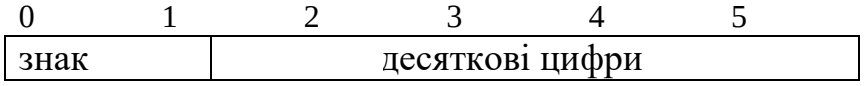

Рисунок 1.2 – Формат десяткових даних навчальної ЕОМ

Старший розряд слова даних використовується для кодування знака: плюс(+) зображується як 0, мінус (-) як 1. Якщо результат арифметичної операції виходить за межі вказаного діапазону, то кажуть, що сталося перепереповнення розрядної сітки. АЛУ в цьому випадку виробляє сигнал переповнення OV = I. Результатом операції ділення є ціла частина частки. Ділення на нуль викликає переповнення.

1.2 Система команд

При розгляді системи команд ЕОМ зазвичай аналізують три аспекта: формати, способи адресації і систему операцій.

1.2.1 Формати команд

Більшість команд навчальної ЕОМ є одноадресними або безадресними, довжиною в одне машинне слово (6 розрядів). Виняток становлять двохсловні команди з безпосередньою адресацією і команда mov, являються двохдресними.

У форматах команд виділяється три поля:

1. два старших розряди [0: 1] визначають код операції СОР;
2. розряд 2 може визначати тип адресації (в одному випадку (формат 5а) він визначає номер регістра);
3. розряди [3: 5] можуть визначати прямий або непрямий адреса пам'яті, номер регістра (в команді mov номера двох регістрів), адреса переходом чи короткий безпосередній операнд. У двохсловних командах безпосередній операнд займає поле [6:11].

Повний список форматів команд показаний на рисунку 1.3, де прийняті наступні позначення:

1. СОР - код операції;
2. ADR - адреса операнда в пам'яті;
3. ADC - адреса переходу;
4. I - безпосередній операнд;
5. R, Rl, R2 - номер регістра;
6. ТА - тип адресації;
7. X - розряд не використовується.

012345

1	COP	X	XXX	
2	COP	TA	ADR	
3	COP	TA	XXR	
3a	COP	TA	X R1 R2	611
4	COP	X	XXX	I
5	COP	X	ADC	
5a	COP	R	ADC	

Рисунок 1.3 – Формати команд навчальної ЕОМ

1.2.2 Способи адресації

У ЕОМ прийнято розрізняти п'ять основних способів адресації: пряма, непряма, безпосередня, відносна, безадресна.

Кожен спосіб має різновиди. У моделі навчальної ЕОМ реалізовані сім способів адресації, наведені в таблиці 1.1.

Таблиця 1.1 – Адресація в командах навчальної ЕОМ

Код ТА	Тип адресації	Виконавча адреса
0	Пряма (реєстрова)	ADR (R)
1	Безпосередня	-
2	Непряма	ОЗП (ADR) [3: 5]
3	Відносна	ADR + RB
4	Побічно-реєстрова	РОН (R) [3: 5]
5	Індексна з постінкрементом	РОН (R) [3: 5], R: = R + 1
6	Індексна з преддкрементом	R: = R-1, РОН (R) [3: 5]

1.2.3 Система операцій

Система команд навчальної ЕОМ включає команди наступних класів:

1. Арифметико-логічні та спеціальні: додавання, віднімання, множення, ділення;

2. Пересилання і завантаження: читання, запис, пересилання (з регістра в регістр), переміщення в стек, витяг з стека, завантаження покажчика стека, завантаження базового регістра;

3. Введення / виведення: введення, висновок;

4. Передачі управління: безумовний і шість умовних переходів, виклик підпрограми, повернення з підпрограми, цикл, програмне пре- ривання, повернення з переривання;

5. Системні: порожня операція, дозволити переривання, заборонити преривання, стоп.

Список команд навчальної ЕОМ наведено в таблицях 1.4 і 1.6.

1.3 Стан і режими ЕОМ

Ядром ОЗП ЕОМ є керуючий автомат (КА), що виробляє сигнали управління, які ініціюють роботу АЛП, РОН, ОЗП і УВВ, передачі інформації між регістрами пристроїв ЕОМ і дії над вмістом регістрів ОЗП.

ЕОМ може знаходитися в одному з двох станів: Зупинка і Робота.

У стані Робота ЕОМ переходить за дією команд Пуск або Крок. Команда Пуск запускає виконання програми, що представляє собою послідовність команд, записаних в ОЗП, в автоматичному режимі до команди hlt або точки зупинки. Програма виконується за командами, починаючи з комірки ОЗП, на яку вказує РС, причому зміна станів об'єктів моделі відображається у вікнах оглядачів.

У стані Зупинка ЕОМ переходить к дії команди Стоп або автоматично в залежності від встановленого режиму роботи.

Команда Крок, в залежності від встановленого режиму роботи, запускає виконання однієї команди або однієї мікрокоманди (якщо встановлений Режим мікрокоманд), після чого переходить в стан Зупинка.

У стані Зупинка заборонений перегляд і модифікація об'єктів моделі: регістрів процесора і РОН, комірок ОЗП, пристроїв введення / виводу. У

процесі модифікації комірок ОЗП і РОН можна вводити дані для програми, в комірку ОЗП - програму в кодах. Крім того, в режимі Зупинка можна змінювати параметри моделі і режими її роботи, вводити або редагувати програму в мнемокодах, асемблірувати мнемокоди, виконувати стандартні операції з файлами.

1.4 Інтерфейс користувача

У програмній моделі навчальної ЕОМ використаний стандартний інтерфейс Windows, реалізований в декількох вікнах.

Основне вікно моделі Модель навчальної ЕОМ містить основне меню і кнопки на панелі управління. В робоче поле вікна виводяться повідомлення про функціонування системи в цілому. Ці повідомлення групуються в файлі logfile.txt (за замовчуванням), зберігаються на диску і можуть бути проаналізіровані після завершення сеансу роботи з моделлю.

Також можуть бути доступними наступні пункти і команди:

1. Файл:
   - Неактивні команди;
   - Вихід.
2. Вид:
   - Показати все;
   - Приховати все;
   - Процесор;
   - Мікрокомандном рівень;
   - Пам'ять;
   - Кеш-пам'ять;
   - Програма;
   - Текст програми.
3. Зовнішні пристрої:
   - Менеджер ВУ;
   - Вікна підключених ВУ;
4. Робота:
   - Пуск;
   - Стоп;
   - Крок;
   - Режим микрокоманд;
   - Кеш-пам'ять;
   - Налаштування.

Команди меню Вид відкривають вікна відповідних оглядачів, описані далі. Менеджер зовнішніх пристроїв дозволяє подключати / відключати зовнішні пристрої, передбачені в системі. Команда виклику менеджера зовнішніх пристроїв виконується при натисканні кнопки на панелі інструментів. Команди меню Робота дозволяють запустити програму автоматично (команда Пуск) або кроком (команда Крок) режимі, зупинити виконання програми в моделі процесора (команда Стоп). Ці команди можуть виконуватися при

натисканні відповідних однойменних кнопок на панелі інструментів основного вікна.

Команда Режим мікрокоманд включає / вимикає мікрокомандний режим роботи процесора, а команда Кеш-пам'ять підключає / відключає в системі модель цього пристрою.

Команда Налаштування відкриває діалогове вікно Параметри системи, яке позволяє встановити затримку реалізації командного циклу (при виконанні програми в автоматичному режимі), а так само встановити параметри файлу Iogfile.txt, сформованого системою і записуємого на диск.

1.4.1 Вікно “Процесор”

Вікно “Процесор” (рисунок 1.4) забезпечує доступ до всіх регістрів і прапорців процесора.

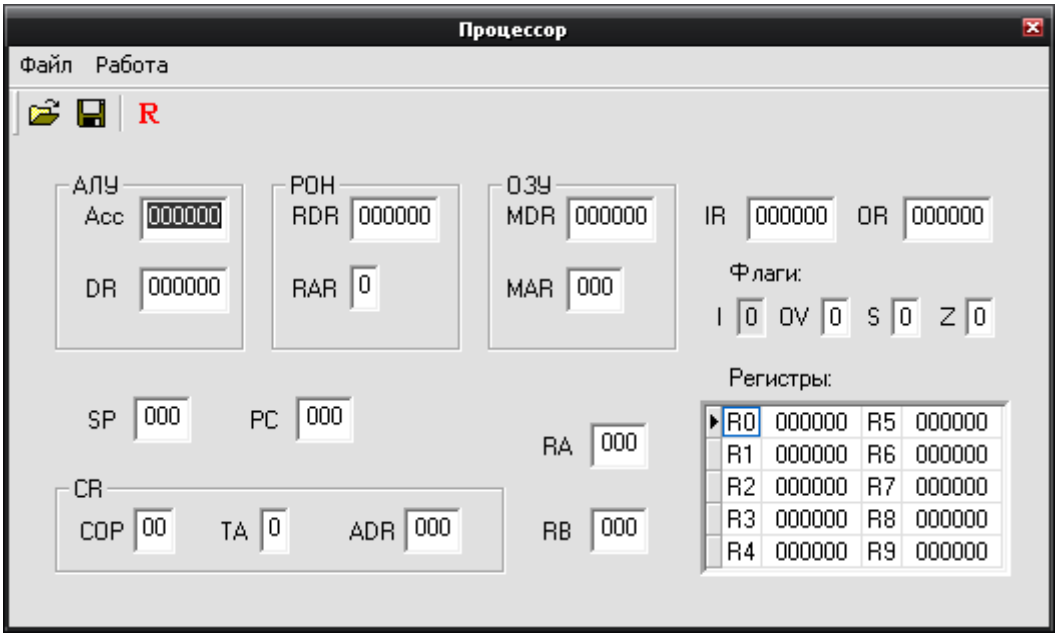

Рисунок 1.4 – Вікно “Процесор”

Програмно-доступні регістри і прапорці:

– Ac - акумулятор;
– PC - лічильник адреси команди, що містить адресу поточної команди;
– SP - покажчик стека, що містить адреса верхівки стека;
– RB - регістр базової адреси, що містить базову адресу;
– RA - регістр адреси, що містить виконавчий адрес при непрямійадресації;
– IR - вхідний регістр;
– OR - вихідний регістр;
– I - прапор дозволу переривань.

Системні регістри і прапорці:

– DR - регістр даних АЛП, що містить другий операнд;
– MDR - регістр даних ОЗП;

– MAR - регістр адреси ОЗП;
– RDR - регістр даних блоку РОН;
– RAR - регістр адреси блоку РОН;
– CR - регістр команд, що містить поля;
– COP - код операції;
– TA - тип адресації;
– ADR - адреса або безпосередній операнд Z - прапор нульового значення Acc;
– S - прапор від'ємного значення Aci;
– OV - прапор переповнення.

Регістри Aci, DR, IR, OR, CR і все комірки ОЗП і РОН мають довжину 6 десяткових розрядів, регістри PC, SP, RA і RB - 3 розряду. У вікні Процесор відображаються поточні значення регістрів і прапорів, причому в стані Зупинка все регістри, включаючи регістри блоку РОН, і прапори (крім прапора I) доступні для безпосереднього редагування.

Елементи управління вікна Процесор включають меню і кнопки, які визивають команди:

– Зберегти;
– Завантажити;
– Reset;
– ResetR0-R9 (тільки команда меню Робота).

Команди Зберегти, Завантажити дозволяють зберегти поточне значення регістрів і прапорів процесора в файлі і відновити стан процесора з файлу. Команда Reset і кнопка R встановлюють всі регістри (в т. Ч. Блок РОН) в початкове (нульове) значення. Вміст елементів пам'яті при цьому не змінюється. Виконувана лише з меню Робота команда ResetR0-R9 очищає тільки регістри блоку РОН.

1.4.2 Вікно “Пам'ять”

Вікно “Пам’ять” (рисунок 1.5) відображає поточний стан комірок ПУ. У цьому вікні допускається редагування вмісту комірок, крім того, передбачено можливість виконання (через меню або за допомогою кнопок панелі інструментів) п'яти команд: Зберегти, Завантажити, Перейти до, Вставити, Прибрати.

Команди Зберегти, Завантажити у всіх вікнах, де вони передбачені, працюють однаково - зберігають у файлі поточний стан об'єкта (в даному випадку пам'яті) і відновлюють цей стан з обраного файлу, причому файл в кожному вікні записується за замовчуванням з характерним для цього вікна розширенням.

Команда Перейти до відкриває діалогове вікно, що дозволяє перейти на задану комірку ПУ.

Команда Прибрати відкриває діалог, в якому вказується діапазон комірок з m поп. Вміст осередків в цьому діапазоні втрачається, а вміст комірок [(i+1): 999] переміщається в сусідні комірки з меншими адресами. Звільнені комірки з

адресами 999, 998, ... заповнюються нулями.

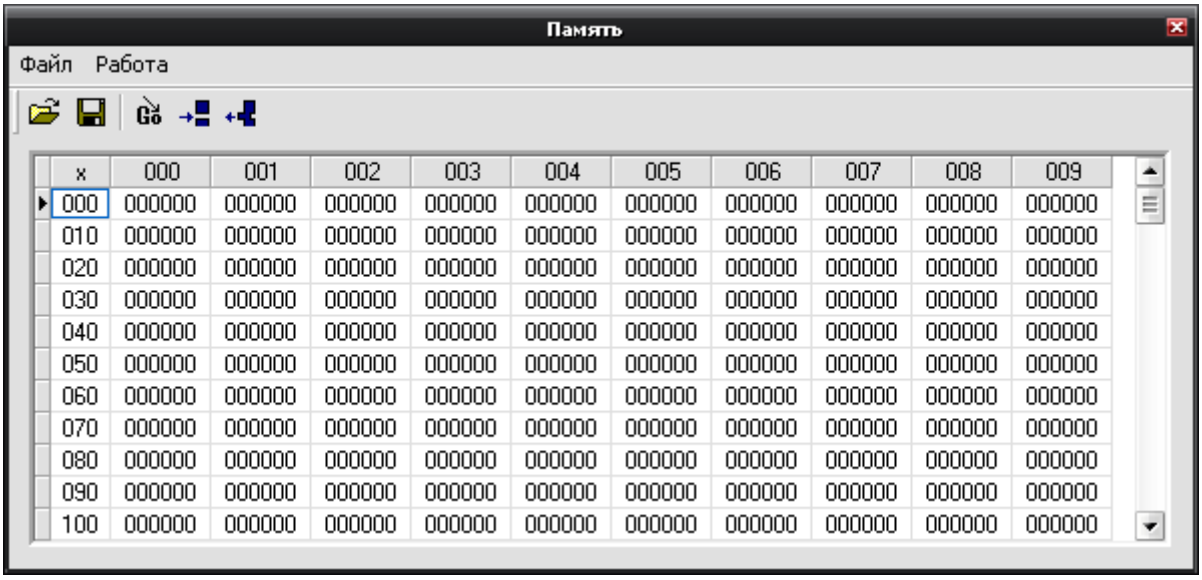

Рисунок 1.5 – Вікно “Пам'ять”

Команда Вставити, що дозволяє задати номером позиції переміщує зміст всіх комірок, починаючи від т-й на п-т позицій в напрямку великих адрес, комірки заданого діапазону [т:п] заповнюються нулями, а вміст останніх осередків пам'яті втрачається.

1.4.3 Вікно “Текст програми”

Вікно “Текст програми” містить стандартне поле текстового редактора, в якому можна редагувати тексти, завантажувати в нього текстові файли і зберігати підготовлений текст у вигляді файлу.

Команди меню Файл:

Нова - Відкриває новий сеанс редагування;

Завантажити - Відкриває стандартний діалог завантаження файлу в вікно редактора;

Зберегти - Зберігає файл під поточним ім'ям;

Зберегти як - Відкриває стандартний діалог збереження файлу;

Вставити - Дозволяє вставити обраний файл в позицію курсора.

Всі перераховані команди, крім останньої, дубльовані кнопкамина панелі інструментів вікна. На тій же панелі присутній ще одна кнопка -Компілювати, яка запускає процедуру асемблірування тексту в поле редактора.

Ту ж процедуру можна запустити з меню Робота. Команда Адреса вставки дозволяє задати адресу комірки ОЗП, починаючи з якої програма буде розміщуватися в пам'яті. За замовчуванням ця адреса прийнятий рівним 0.Нижче області редагування в рядок стану виводиться позиція поточного рядка редактора - номер рядка, в якій знаходиться курсор.

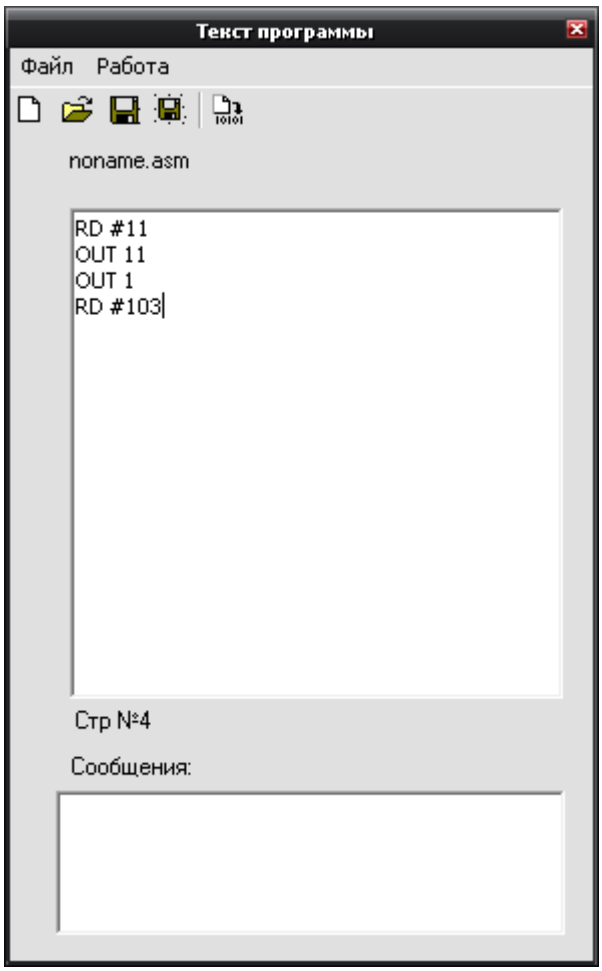

Рисунок 1.6 – Вікно “Текст програми”

У разі виявлення синтаксичних помилок в тексті програми діагностичні повідомлення процесу компіляції виводяться у вікно повідомлень і запис в пам'ять кодів (навіть безпомилкового початкового фрагмента програми) не проводиться.

Після виправлення помилок і повторної компіляції видається повідомлення про відсутність помилок, про розташування та розмір області пам'яті, зайнятої під асемблірованную програму.

Набір тексту програми проводиться за стандартними правилами мови асемблера. У кожному рядку може міститися мітка, одна команда з коментарями. Мітка відділяється від команди двокрапкою, символи після знака "крапка з комою" до кінця рядка ігноруються компілятором і можуть розглядатися як коментарі. Рядок може починатися з;, отже, зберігати тільки коментарі.

1.4.4 Вікно “Програма”

Вікно “Програма” (рисунок 1.7) відображає таблицю, що має 300 рядків і 4 стовбчики. Кожен рядок таблиці відповідає дизасемблювати комірку ПУ. Другий стовпчик містить адресу комірки ОЗП, третій – дизасемблірування мнемокоду, четвертий - машинний код команди. У першому стовпціможе

міститися покажчик -> на поточну команду (поточне значення PC) і точка зупинки - червона заливка комірки.

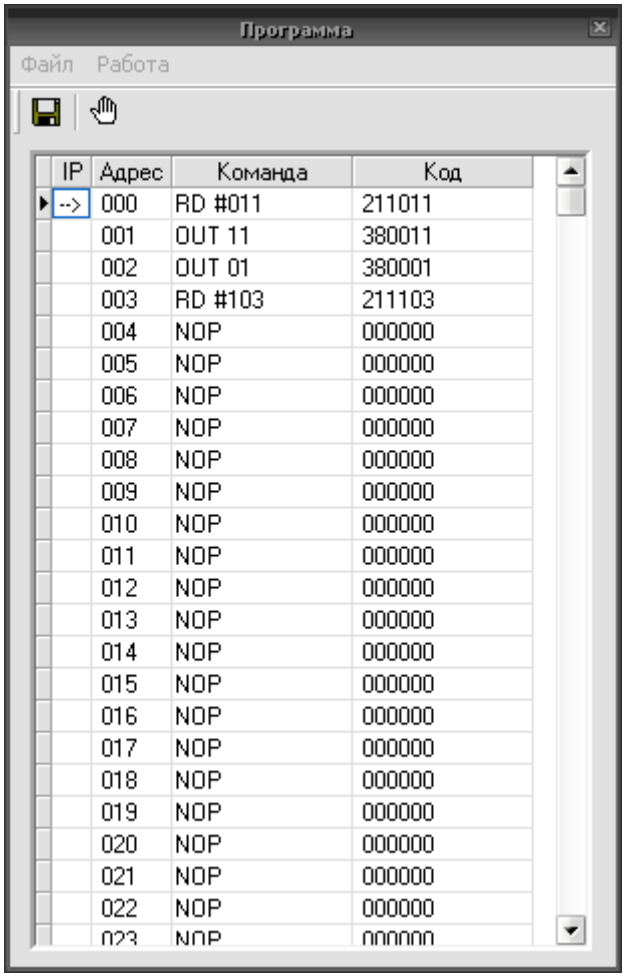

Рисунок 1.7 – Вікно “Програма”

Вікно Програма дозволяє спостерігати процес проходження програми. У цьому вікні нічого не можна редагувати. Органи управління вікна дозволяють зберегти вміст вікна у вигляді текстового файлу, вибрати початкову адресу області ОЗП, яка буде аналізувати код (розмір області постійний -300 комірок), а також встановити / зняти точку зупину. Останнє можна зробити трьома способами: командою Точка зупину з меню Робота, кнопкою на панелі інструментів або подвійним клацанням миші в першій клітинці відповідної рядка.

Характерно, що прочитати в це вікно нічого не можна. Збережений текстовийasm-файл можна завантажити у вікно Текст програми, асемблірувати його і тоді аналізувати код значення заданої області пам'яті автоматично з'явиться у вікні Програма. Таку процедуру зручно використовувати, якщо програма спочатку пишеться або редагується безпосередньо в пам'яті в машинних кодах. Початкова адреса області дизасемблірування задається в діалозі командою Початкова адреса меню Робота команд , яка встановлюється командою Режим мікрокоманд меню Робота.В це вікновиводиться мнемокод виконуваної команди, список мікрокоманд, її

реалізують, та покажчик на поточну виконувану мікрокоманду.

Кроковий режим виконання програми або запуск програми в автономному режимі з затримкою командного циклу дозволяє спостерігати процес виконання програми на рівні мікрокоманд.

Якщо відкрити вікно мікрокомандний рівень, шляхом відмови від встановлення режиму мікрокоманд в меню Робота, то після початку виконання програми в режимі Крок (або в автоматичному режимі) в рядку повідомлень вікна буде видано повідомлення "Режим мікрокоманд неактивний".

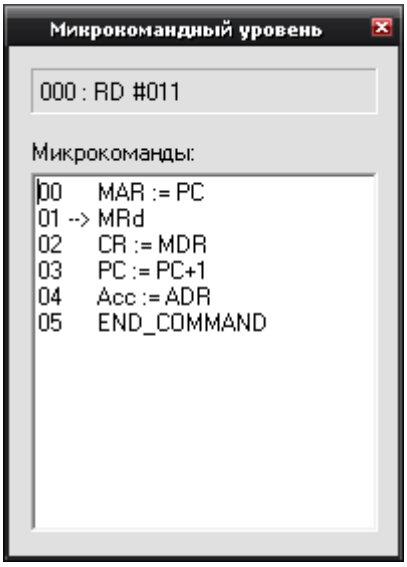

Рисунок 1.8 – Вікно «мікрокомандний рівень»

1.4.5 Вікно Кеш-пам’ять

Вікно Кеш-пам'ять використовується в режимі з підключеною кеш-пам'яттю.

Детальніше дивіться про цей режим в розділі 1.8.

1.5 Зовнішні пристрої

Моделі зовнішніх пристроїв (ЗП), що використовуються в описуваній системі, реалізовані за єдиним принципом. З точки зору процесора вони представляють собою ряд програмно-доступних регістрів, що лежать в адресному просторі введення / виводу. Розмір регістрів ЗП збігається з розміром комірок пам’яті регістрів даних процесора - шість десяткових розрядів.

Доступ до регістрів ЗП здійснюється по командам in aa, out aa, де aa - двухразрядний десятковий адреса регістра ЗП. Таким чином, загальний обсяг адресного простору введення / виведення становить 100 адрес. Слід помнити, що адресні простори пам'яті і введення / виводу в цій моделі розділені.

Різні ЗП містять різну кількість програмно-доступних регістрів кожному, з яких відповідає свою адресу, причому нумерація адрес всіх ЗП починається з 0. При створенні ЗП йому ставиться в відповідність базовий адрес в просторі

введення / виводу, і всі адреси його регістрів стають зміщеними щодо цього базового адресу.

Якщо в системі створюються кілька ЗП, то їх базові адреси слід вибирати з урахуванням величини адресного простору, займаного цими пристроями, виключаючи накладення адрес.

Якщо ЗП здатне формувати запит на переривання, то при створенні йому ставиться в відповідність вектор переривання - десяткове число. Різним ЗП повинні призначатися різні вектори переривань.

Програмна модель навчальної ЕОМ комплектується набором зовнішніх пристроїв, що включає:

1. контролер клавіатури;
2. дисплей;
3. блок таймерів;
4. тоногенератор.

Цим пристроям за замовчуванням присвоєні параметри, перераховані в таблиці 1.2.

Таблиця 1.2 – Параметри зовнішніх пристроїв

Зовнішній пристрій	Базова адреса	Адреси регістрів	Вектор переривання
Контролер клавіатури	0	0, 1, 2	0
Дисплей	10	0, 1, 2, 3	Ні
Блок таймерів	20	0,1,2,3,4,5,6	2
Тоногенератор	30	0,1	Ні

При створенні пристроїв користувач може змінити призначені за замовчуванням базова адреса і вектор переривання.

У описуваної версії системи не передбачена можливість підключення в систему декількох однакових пристроїв.

Більшість зовнішніх пристроїв містить регістри управління CR і стану SR, причому зазвичай регістри CR доступні тільки по запису, а SR - читання.

Регістр CR містить прапори і поля, що визначають режими роботи ВУ, а SRпрапори, що відображають поточний стан ВУ. Прапори SR встановлюються апаратно, але скидаються програмно (або по зовнішньому сигналу). Поля і прапори CR встановлюються і скидаються програмно під час запису коду данних в регістр CR або спеціальними командами.

Контролер ВУ інтерпретує код, записується за адресою CR як команду, якщо третій розряд цього коду дорівнює 1, або як записуються в CR дані, якщо третій розряд дорівнює 0. У разі отримання командного слова запис в регістр CR не проводиться, а п'ятий розряд слова розглядається як код операції.

1.5.1 Контролер клавіатури

Контролер клавіатури (рисунок 1.9) являє собою модель зовнішнього пристрою, що приймає ASCII-коди від клавіатури ПЕОМ. Символи поміщаються послідовно в буфер символів, розмір якого встановлений рівним 50 символам, і відображаються у вікні оглядача.До складу контролера клавіатури входять три програмно-доступних регістра:

– DR (адреса 0) - регістр даних;
– CR (адреса 1) - регістр управління, визначає режими роботи контролера і містить наступні прапори:
– E - прапор дозволу прийому кодів в буфер;
– I - прапор дозволу переривання;
– S - прапор режиму посимвольного введення.
– SR (адреса 2) - регістр стану, містить два прапори:
– Err - прапор помилки;
– Rd - прапор готовності.

Регістр даних DR доступний тільки для читання, через нього зчитуються ASCII-коди з буфера, причому порядок читання кодів з буфера відповідає порядку їх запису в буфер - кожне читання за адресою 0 автоматично переміщує покажчик читання буфера. У кожен момент часу DR містить код символу за адресою покажчика читання буфера.

Прапори регістра управління CR встановлюються і скидаються програмно. Прапор E, будучи встановленим, дозволяє прийом кодів в буфер. При E =O контролер ігнорує натискання на клавіатурі, прийом кодів у буфер не проводиться. На зчитування кодів з буфера прапор E впливу не робить.

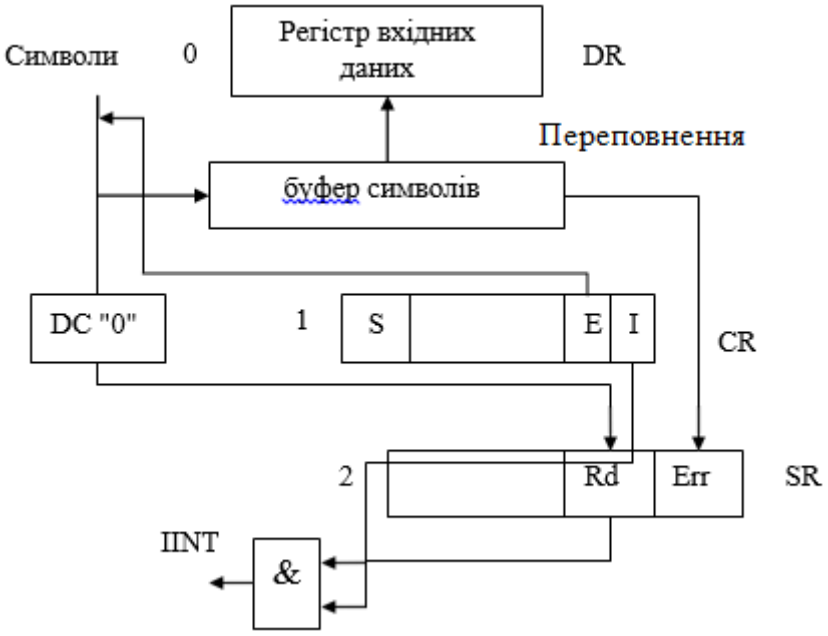

Рисунок 1.9 – Контролер клавіатури

Прапор I, будучи встановленим, дозволяє при певних умовах формування

контролером запиту на переривання. При I = 0 запит на переривання не формується.

Прапор S = 1 встановлює режим посимвольного введення, інакше контроллер працює в звичайному режимі. Прапор S встановлюється і скидається програмно, крім того, S скидається при натисканні кнопки Очистити буфер у вікні Контролер клавіатури.

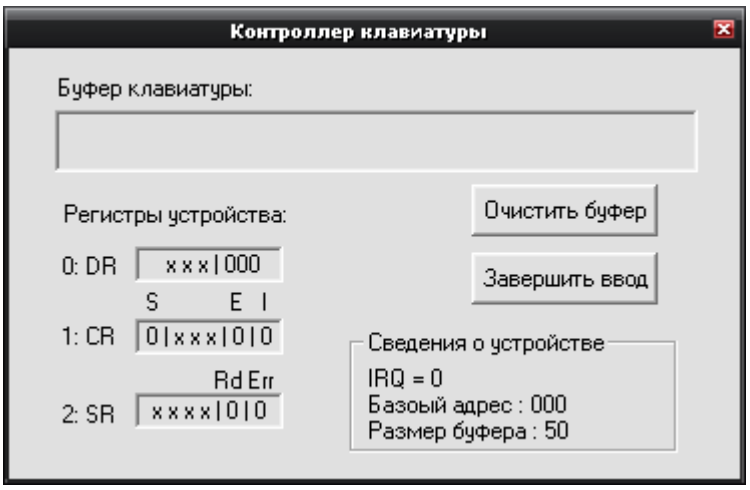

Рисунок 1.10 – Вікно оглядача контролера клавіатури

Умови формування запиту на переривання визначаються, з одногобоку, значенням прапора дозволу переривання I, з іншого - режимом роботи контролера. У режимі посимвольного введення запит на переривання формується після введення кожного символу (зрозуміло, при 1 = 1), в звичайному режимі запит буде сформований після закінчення набору рядки.

Завершити набір рядки можна, натиснувши кнопку Завершити введення в вікні Контролер клавіатури. При цьому встановлюється прапор готовності Rd (від англ. Ready) в регістрі стану SR. Прапор помилки Err (від англ. Error) в тому ж регістрі встановлюється при спробі введення в буфер 51 -го символу. Введення 51-го і всіх наступних символів блокується.

Скидання прапора Rd здійснюється автоматично при читанні з регістра DR, прапор Err скидається програмно. Крім того, обидва ці прапора зкидуються при натисканні кнопки Очистити буфер у вікні Контролер клавіатури; одночасно зі скиданням прапорів проводиться очищення буфера весь буфер заповнюється кодами 00h, і покажчики запису і читання встановлюються на початок буфера.

Для програмного управління контролером передбачений ряд командних слів. Всі команди виконуються при записі за адресою регістра управління CR кодів з 1 в третьому розряді.

Контролер клавіатури інтерпретує такі командні слова: xxxlOl - очистити буфер (дія команди еквівалентно натисканню кнопки **- Очистити** буфер);

– xxx102 - скинути прапор Err в регістрі SR;

– xxx103 - встановити прапор S в регістрі CR;

– xxx104 - скинути прапор S в регістрі CR.

Якщо за адресою 1 зробити запис числа xxx0nn, то відбудеться зміна 4-го і 5-го розрядів регістра CR.

1.5.2 Дисплей

Дисплей (рисунок 1.11) являє собою модель зовнішнього пристрою, що реалізує функції символьного дисплея. На дисплеї можуть відображатися символи, що задаються ASCII-кодами, які надходять на його регістр даних. Дисплей включає:

– Відеопам'ять об'ємом 128 слів (ОЗП дисплея);
– Символьний екран розміром 8 рядків по 16 символів в рядку;
– Чотири програмно-доступних регістра:
– DR (адреса 0) - регістр даних;
– CR (адреса 1) - регістр управління;
– SR (адреса 2) - регістр стану;
– AR (адреса 3) - регістр адреси.

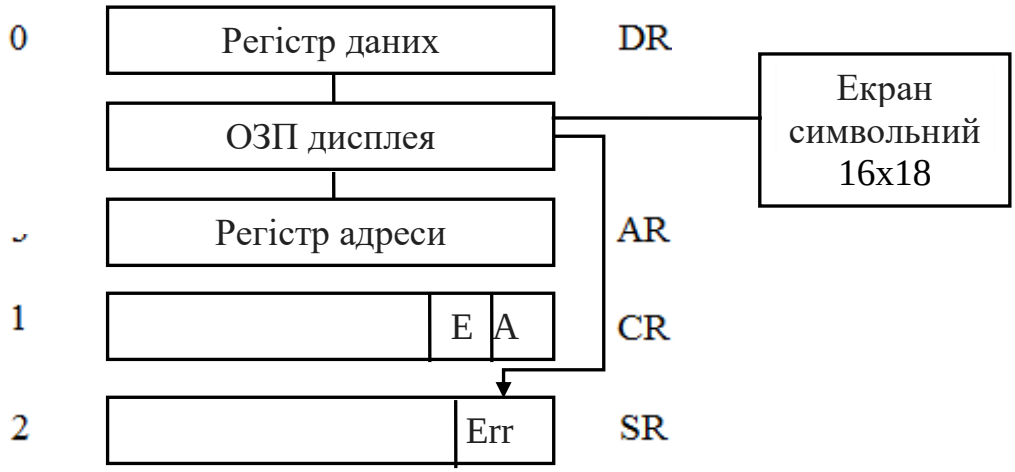

Рисунок 1.11 – Контролер дисплея

Через регістри адреси AR і даних DR по запису і читанню здійснюється доступ до комірок відеопам'яті. При зверненні до регістру DR по запису вміст акумулятора записується в DR і в комірку відеопам'яті, адреса якої встановлено в регістрі AR. Регістр управління CR доступний тільки за попереднім записом і містить в 4-м і 5-м розрядах відповідно два прапори:

– E – прапор дозволу роботи дисплея; при E = 0 запис в регістри AR і DR блокується;

– A – прапор автоінкремента адреси; при A = 1 вміст AR автоматично збільшується на 1 після будь-якого звернення до регістру DR- по запису або читання.

Змінити значення цих прапорів можна, якщо записати за адресою CR (за замовчуванням - 11) код xxxOii, при цьому зміна 4-го і 5-го розрядів регістра CR відбудеться відповідно до виразу (1).

Для програмного управління дисплеєм передбачені дві команди, коди яких повинні записуватися за адресою регістра CR, причому в третьому розряді командних слів обов'язково повинна бути 1:

— xxxlOl – очистити дисплей (дія команди еквівалентно натискання кнопки Очистити в вікні Дисплей), при цьому очищається відеопам'ять (в кожну комірку записується код пробілу- 032), встановлюється в 000 регістр адреси AR і скидаються прапори помилки Err і автоінкремента A;

— xxx 102 – скинути прапор помилки Err.

Регістр стану SR доступний тільки з читання і містить єдиний прапор (в п'ятому розряді) помилки Err. Цей прапор встановлюється апаратно при спробі записати в регістр адреси число, більше 127, причому як в режимі прямого запису в AR, так і в режимі автоінкремента після звернення за адресою 127. Скидається прапор Err програмно або при натисканні кнопки Очистити в вікні Дисплей.

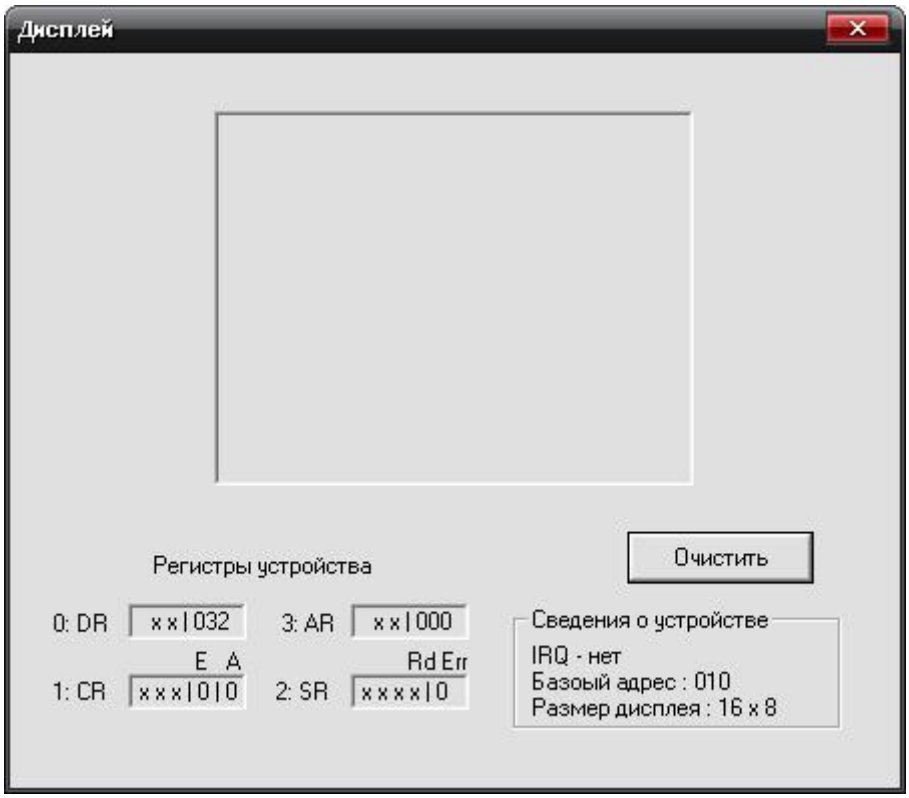

Рисунок 1.12 – Вікно оглядача контролера дисплея

1.5.3 Блок таймерів

Блок таймерів (рисунок 1.13) включає в себе три однотипних канали, кожен з яких містить:

— п`ятирозрядний десятковий реверсивний лічильник T, на вхід якого надходять мітки часу (таймер);

— Програмований переддільник D;

— Регістр управління таймером CTR;

— Прапор переповнення таймера FT.

Регістри таймерів Т доступні за попереднім записом і читанням (адреси 1, 3, 5 відповідно для Т1, Т2, Т3). Програма в будь-який момент може вважати поточний зміст таймера або записати в нього нове значення.

На входи переддільника надходить загальні для всіх каналів мітки часу CLK з періодом 1 мс. Переддільника в кожному каналі програмуються незалежно, тому таймери можуть працювати з різною частотою. Регістри управління CTR доступні за попереднім записом і читанням (адреси 2, 4, 6) і містять такі поля:

— Т (розряд 5) - прапор включення таймера;

— EI (розряд 4) - прапор дозволу формування запиту на переривання при переповненні таймера;

— I/D (розряд 3) - напрям рахунку (інкремент/декремент), при I/D = 0 таймер працює на складання, при I/D = 1 - на віднімання;

— *к* (розряди [1: 2]) - коефіцієнт ділення переддільника (від 1 до 99).

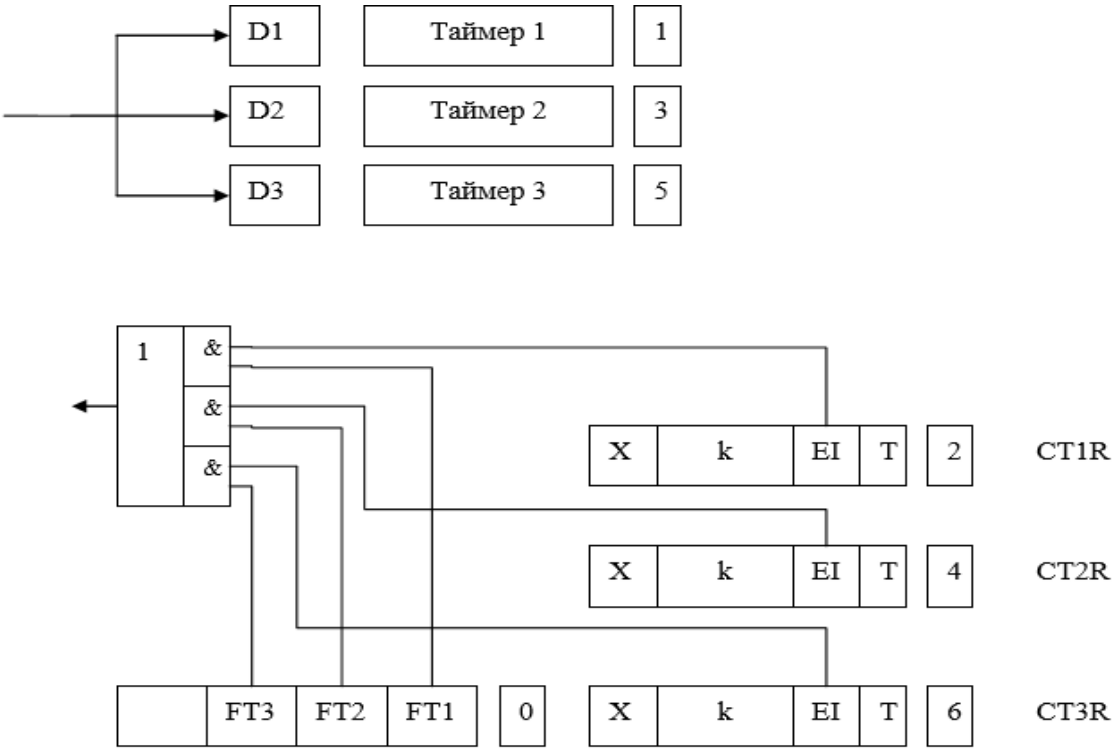

Рисунок 1.13 – Блок таймерів

Прапори переповнення таймерів зібрані в один регістр — доступний тільки з читання регістр стану SR, що має адресу 0. Розряди регістра (5, 4 і 3 для Т1, Т2, Т3 відповідно) встановлюються в 1 при переповненні відповідного таймера. Для таймера, що працює на додавання, переповнення настає при переході його стану з 99 999 в 0, для віднімаючого таймеру – перехід з 0 в 99999.

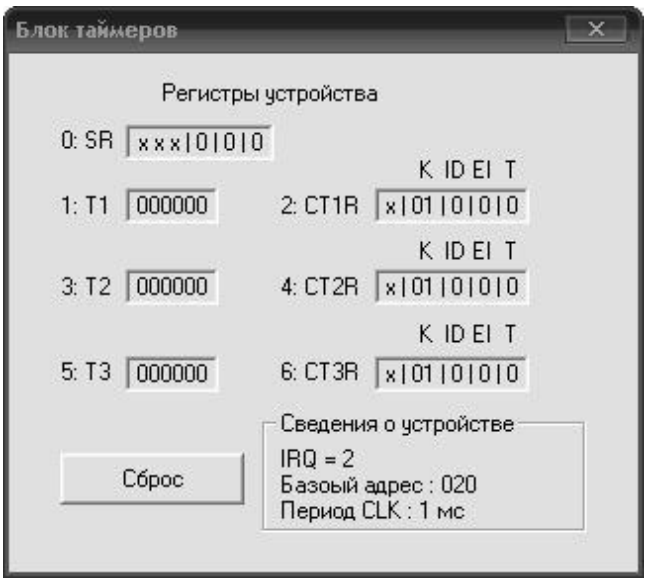

Рисунок 1.14 – Вікно оглядача блоку таймерів

У вікні оглядача (рисунок 1.14) передбачена кнопка “Сброс”, натиснувши яку скидає в 0 всі регістри блоку таймерів, крім CTR, які встановлюються в стан 001000. Таким чином, всі три таймера обнуляються, переключаються в режим інкременту, припиняється рахунок, забороняються переривання, скидаються прапори переповнення і встановлюються коефіцієнти розподілу переддільника рівними 01. Програмне управління режимами блоку таймерів здійснюється шляхом запису в регістри CTR відповідних кодів. Запис за адресою SR числа з 1 в третьому розряді інтерпретується блоком таймерів як команда, причому молодші розряди цього числа визначають код команди:

— xxx100 – загальне скидання (еквівалентна натискання кнопки “Сброс” у вікні оглядача);

— xxx101 – скидання прапора переповнення таймера FT1;

— xxx102 – скидання прапора переповнення таймера FT2;

— xxx103 – скидання прапора переповнення таймера FT3.

1.5.4 Тоногенератор

Модель цього простого зовнішнього пристрою не має власного оглядача, містить всього два регістри, доступних тільки для запису:

– FR (адреса 0) – регістр частоти звучання (Гц);

– LR (адреса 1) – регістр тривалості звучання (мс).

За замовчуванням базова адреса тоногенератора - 30. Спочатку слід записати в FR необхідну частоту тону в герцах, потім в LR - тривалість звучання в мілісекундах. Запис числа за адресою регістра LR одночасно є командою на початок звучання.

1.6 Підсистема переривань

У моделі навчальної ЕОМ передбачений механізм векторних зовнішніх переривань. Зовнішні пристрої формують запити на переривання, що надходять

на входи контролера переривань. При підключенні ВУ, здатного формувати запит на переривання, йому ставиться в відповідність номер входу контролера переривань - вектор переривання, що приймає значення в діапазоні 0 - 9.

Контролер передає вектор, відповідний запит, процесору, який починає процедуру обслуговування переривання.

Кожному з можливих в системі переривань повинен відповідати т. н. обробник переривання - підпрограма, що викликається при виникненні події конкретного переривання.

Механізм переривань, реалізований в моделі навчальної ЕОМ, підтримує таблицю векторів переривань, яка створюється в оперативній пам'яті моделлю операційної системи (якщо вона використовується) або безпосередньо користувачем.

Номер рядка таблиці відповідає вектору переривання, а елемент таблиці - комірка пам'яті, в трьох молодших розрядах якої розміщується початкова адреса підпрограми, яка обслуговує переривання з цим вектором.

Таблиця переривань в розглянутій моделі жорстко фіксована - вона займає комірку пам'яті з адресами 100-109. Таким чином, адреса обробника з вектором 0 повинен розташовуватися в комірці 100, з вектором 2 - в комірці 102. При роботі з перериваннями не рекомендується використовувати комірки 100-109 для інших цілей.

Процесор починає обробку переривання (якщо вони дозволені), завершивши поточну команду. При цьому він:

1. Отримує від контролера вектор переривання.

2. Формує та поміщає в верхівку стека слово, три молодших розряди ([3: 5]) у яких поточне значення PC (адреса повернення з переривання), а розряди [1: 2] зберігають десятковий еквівалент шістнадцятирічної цифри, що визначає значення вектора прапорів (1, OV, S, Z). Наприклад, якщо 1 = 1, OV = 0, S = 1, Z = 1, то в розряди [1: 2] запишеться число 1110 = 10112.

3. Скидає в 0 прапор дозволу переривання 1.

4. Витягує з таблиці векторів переривань адреса обробника, відповідний обслуговуваний вектор, і поміщає його в PC, здійснюючи тим самим перехід на підпрограму обробника переривання.

Таким чином, виклик обробника переривання, на відміну від виклику підпрограми, пов'язаний з приміщенням в стек не тільки адреси повернення, а й поточного значення вектора прапорів. Тому останньою командою підпрограми обробника повинна бути команда iret, яка не тільки повертає в PC три молодші розряду комірки - верхівки стека (як ret), але і відновлює ті значення прапорів, які були в момент переходу на обробник переривання.

Не всяка подія, яка може викликати переривання, призводить до переривання поточної програми. До складу процесора входить програмно доступний прапор I дозволу переривання. При I = 0 процесор не реагує на запити переривань. Після скидання процесора прапор I так же скинуто і все переривання заборонені. Для того щоб дозволити переривання, слід в програмі виконати команду ei (від англ. Enable interrupt).

Вище зазначалося, що при переході на обробник переривання прапор I автоматично скидається, в цьому випадку перервати обслуговування одного переривання іншим перериванням не можна. За командою iret значення прапорів відновлюється, в т. Ч. Знову встановлюється 1 = 1, отже, в основній програмі переривання знову дозволені.

Якщо потрібно дозволити інші переривання в обробнику переривання, досить в ньому виконати команду ei. Контролер переривань і процесор на апаратному рівні блокують спроби запустити переривання, якщо його обробник почав, але не завершив роботу.

Таким чином, прапор I дозволяє або забороняє всі переривання системи. Якщо потрібно вибірково дозволити деяку підмножину переривань, використовуються програмно-доступні прапори дозволу переривань безпосередньо на зовнішніх пристроях.

Як правило, кожне зовнішнє пристрій, який може викликати переривання, містить в складі своїх регістрів розряд прапора дозволу переривання (див. Формат регістрів CR і CTR на рис. 9, 13), за замовчуванням встановлений в 0. Якщо залишити цей прапор в нулі, то зовнішнього пристрою забороняється формувати запит контролеру переривань.

Іноді буває зручно (наприклад, в режимі налагодження) мати можливість викликати обробник переривання безпосередньо з програми. Якщо використовувати для цих цілей команду call, яка поміщає в стек тільки адреса повернення, то команда iret, розміщена останньої в обробнику, може спотворити значення прапорів (всі вони будуть скинуті в 0, т. К. Команда call формує тільки три молодші розряду комірки верхівки стека, залишаючи інші розряди в 000).

Тому в системах команд багатьох ЕОМ, в т. ч. та нашої моделі, є команди виклику переривань - int n (в нашій моделі n є {0, 1, ..., 9}), де n- вектор переривання. Процесор, виконуючи команду int n, виробляє ті ж дії, що і при обробці переривання з вектором n.

Характерно, що за допомогою команди int n можна викликати обробник переривання навіть в тому випадку, коли прапор дозволу переривання 1 скинутий.

1.7 Програмна модель кеш-пам'яті

До описаної в розд. 1 програмної моделі навчальної ЕОМ може бути підключена програмна модель кеш-пам'яті. Кеш-пам'ять містить N комірок (в моделі N може вибиратися з безлічі {4, 8, 16, 32}), кожна з яких включає трьохрозрядне поле тегу (адреси ОЗП), шестирозрядне поле даних і три однобітових ознаки (прапора):

— Z – ознака зайнятості комірки;

— U – ознака використання;

— W – ознака запису в комірку.

Таким чином, кожна комірка кеш-пам'яті може дублювати одну будь-яку комірку ОЗП, причому відзначається її зайнятість (на початку роботи моделі

все комірки кеш-пам'яті вільні, VZi = 0), факт запису інформації в комірку під час перебування її в кеш-пам'яті, а також використання комірки (т. е. будь-яке звернення до неї).

Поточний стан кеш-пам'яті відображається на екрані в окремому вікні в формі таблиці, причому кількість рядків відповідає обраному числу комірок кеш. Стовпці таблиці визначають вміст полів комірок, наприклад, так, як показано в табл. 1.3.

Таблиця 1.3 – Приклад поточного стану кеш-пам'яті

	Теги	Дані	Z	U	W
1	012	220152	1	0	0
2	013	211003	1	1	0
3	050	000025	1	1	1
4	000	000000	0	0	0

Для налаштування параметрів кеш-пам'яті можна скористатися діалоговим вікном кеш-пам'ять, що викликається командою Вид | Кеш-пам'ять, а потім натиснути першу кнопку на панелі інструментів відкритого вікна. Після цих дій з'явиться діалогове вікно Параметри кеш-пам'яті, що дозволяє вибрати розмір кеш-пам'яті, спосіб запису в неї інформації та алгоритм заміщення комірок.

Нагадаємо, що при наскрізний запису при кеш-попаданні в процесорних циклах запису здійснюється запис як в комірку кеш-пам'яті, так і в комірку ОЗП, а при зворотному запису - тільки в комірку кеш-пам'яті, причому ця комірка відзначається бітом запису {Wt: = 1). При очищенні комірок, зазначених бітом запису, необхідно переписати змінене значення нуля даних в відповідному полі ОЗП.

При кеш-промаху слід помістити в кеш-пам'ять, певна комірка яка адресується процесором. При наявності вільних комірок кеш-пам'яті потрібне слово поміщається в одну з них (в порядку черги). При відсутності вільних комірок слід відшукати комірку кеш-пам'яті, вміст якої можна видалити, записавши на його місце необхідні дані (команду). Пошук такої комірки здійснюється з використанням алгоритму заміщення рядків. У моделі реалізовані три різних алгоритма заміщення рядків:

— Випадкове заміщення, при реалізації якого номер комірки кеш-памяті вибирається випадковим чином;

— Черга, при якій вибір змінюваній комірки визначається часом перебування її в кеш-пам'яті;

— Біт використання, випадковий вибір здійснюється тільки з тих комірок, які мають нульове значення прапора використання.

Нагадаємо, що біт використання встановлюється в 1 при будь-якому зверненні до комірки, однак, як тільки всі біти U_i, встановляться в 1, всі вони тут же скидаються в 0, так що в кеш завжди комірки розбиті на два непересічних підмножини за значенням біта U, звернення до яких відбулося

відносно недавно (після останнього скидання вектора U) мають значення U = 1, інші зі значенням U = 0 є "кандидатами на видалення" при використанні алгоритму заміщення "біт використання".

Якщо в параметрах кеш-пам'яті встановлений прапор "з урахуванням біта запису", то всі три алгоритму заміщення здійснюють пошук "кандидата на видалення" перш за все серед тих комірок, ознака запису яких не встановлено, а при відсутності таких комірок (що вкрай малоймовірно) - - серед усіх комірок кешпамяті. При знятому прапорі "з урахуванням біта записи" пошук здійснюється за всіма комірками кеш-пам'яті без урахування значення W.

Оцінка ефективності роботи системи з кеш-пам'яттю визначається числом кеш-влучень по відношенню до загальної кількості звернень до пам'яті. З огляду на різницю в алгоритмах запису в режимах наскрізною і зворотного запису, ефективність використання кеш-пам'яті обчислюється за виразами (1.1) і (1.2) (відповідно для наскрізною і зворотного запису):

$$K = \frac{S_{\text{к}} - S_{\text{к}_w}}{S_{\text{о}}}, \qquad (1.1)$$

$$K = \frac{S_{\text{к}} - S^{i}_{\text{к}_w}}{S_{\text{о}}}, \qquad (1.2)$$

- Де: K – коефіцієнт ефективності роботи кеш-пам'яті;
- S_0 – загальна кількість звернень до пам'яті;
- S_k – число кеш-влучень;
- S_{kw} – число наскрізних записів при кеш-попаданні (в режимі наскрізний запису);
- $S_{k_w}^{\ i}$ – число зворотних записів (в режимі зворотного запису).

1.8 Характеристики і параметри моделі навчальної ЕОМ

У даному розділі представлені таблиці з характеристиками і параметрами моделі навчальної ЕОМ для використання при виконанні лабораторних робіт.

Таблиця 1.4 – Команди навчальної ЕОМ

Мл. Ст.	0	1	2	3	4
0	NOP	JMP		MOV	
1	IN	JZ	RD	RD	RDI
2	OUT	JNZ	WR	WR	
3	IRET	JS	ADD	ADD	ADI
4	WRRB	JNS	SUB	SUB	SBI
5	WRSP	JO	MUL	MUL	MULI
6	PUSH	JNO	DIV	DIV	DIVI
7	POP	JRNZ		IN	

8	RET	INT	EI	OUT	
9	HLT	CALL	DI		

Таблиця 1.5 – Типи адресації, їх коди та позначення

Позначення	**Код**	**Тип адресації**	**Приклад команди**
	0	Пряма (реєстрова)	ADD 23 (ADD R3)
#	1	Безпосередня	ADD #33
@	2	Непряма	ADD @33
[ ]	3	Відносна	ADD [33]
@R	4	Побічно-реєстрова	ADD @R3
@R+	5	Індексна з постінкрементом	ADD @R3+
-@R	6	Індексна з преддекрементом	ADD -@R3

У таблиці 1.5 прийняті наступні позначення:

— DD - дані, що формуються командою в якості (другого) операнда: прямо або побічна адресується комірка пам'яті чи трьохрозрядний безпосередній операнд;

— R * - вміст регістра або побічно адресується через регістр комірки пам'яті;

— ADR * - два молодших розряди ADR поля регістра CR;

— V - адреса пам'яті, відповідний вектору переривання;

— M (*) - елемент пам'яті, прямо або побічно адресується в команді;

— I - п'ятирозрядний безпосередній операнд зі знаком.

Таблиця 1.6 – Система команд навчальної ЕОМ

КОП	Мне-мокод	Назва	Дія
1	2	3	4
00	NOP	Порожня операція	Відсутня
01	IN	Введення	Acc←IR
02	OUT	Вивід	OR←Acc
03	IRET	Повернення з переривання	FLAGS.PC←M(SP); INC(SP)
04	WRRB	Завантаження RB	RB ← CR[ADR]
05	WRSP	Завантаження SP	SP ← CR[ADR]
06	PUSH	Помістити в стек	DEC(SP); M(SP)←R
07	POP	Витягти з стека	R→M(SP); INC(SP)
08	RET	Повернення	PC→ M(SP); INC(SP)
09	HTL	Стоп	Кінець командних циклів
10	JMP	Безумовний перехід	PC←CR[ADR]
11	JZ	Перехід, якщо 0	if Acc=0 then PC←CR[ADR]
12	JNZ	Перехід, якщо не 0	if Acc≠0 then PC←CR[ADR]
13	JS	Перехід, якщо негативно	if Acc<0 then PC←CR[ADR]
14	JNS	Перехід, якщо позитивно	if Acc≥0 then PC←CR[ADR]

15	JO	Перехід, якщо переповнення	if |Acc|>99999 then PC←CR[ADR]
16	JNO	Перехід, якщо немає переповнення	if |Acc|≤99999 then PC←CR[ADR]
17	JRNZ	Цикл	DEC(R);if R>0 then PC←CR[ADR]
18	INT	Програмне переривання	DEC(SP); M(SP)← FLAGS.PC; PC←M(V)
19	CALL	Виклик підпрограми	DEC(SP); M(SP)← PC; PC←CR(ADR)
20	------		
21	RD	Читання	Acc←DD
22	WR	Запис	M(*)←Acc
23	ADD	Додавання	Acc←Acc+DD
24	SUB	Віднімання	Acc←Acc-DD
25	MUL	Множення	Acc←Acc*DD
26	DIV	Ділення	Acc←Acc/DD
27	------		
28	EI	Дозволити переривання	IF←1
29	DI	Заборонити переривання	IF←0
30	MOV	Пересилання	R1←R2
31	RD	Читання	Acc←R*
32	WR	Запис	R*←Acc
33	ADD	Додавання	Acc←Acc+R*
34	SUB	Віднімання	Acc←Acc-R*
35	MUL	Множення	Acc←Acc*R*
36	DIV	Ділення	Acc←Acc/R*
37	IN	Введення	Acc←ВУ(CR[ADR*])
38	OUT	Вивід	ВУ(CR[ADR*])←Acc
39	------		
40	------		
41	RDI	Читання	Acc←I
42	------		
43	ADI	Додавання	Acc←Acc+I
44	SBI	Віднімання	Acc←Acc-I
45	MULI	Множення	Acc←Acc*I
46	DIVI	Ділення	Acc←Acc/I

Таблиця 1.7 – Таблиця ASCII кодів (фрагмент)

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0				0	@	P	‘	p					A	P	a	p
1			!	1	A	Q	a	q					Б	C	б	c
2			“	2	B	R	b	r					B	T	в	т

3			#	3	C	S	c	s					Г	У	г	у
4			$	4	D	T	d	t					Д	Ф	д	ф
5			%	5	E	U	e	u					Е	Х	е	х
6			&	6	F	V	f	v					Ж	Ц	ж	ц
7			‘	7	G	W	q	w					З	Ч	з	ч
8			(	8	H	X	h	x					И	Ш	и	ш
9			)	9	I	Y	i	y					Й	Щ	й	щ
A			*	:	J	Z	j	z					К	Ъ	к	ъ
B			+	;	K	[	k	{					Л	Ы	л	ы
C			,	<	L		l	|					М	Ь	м	ь
D			-	=	M	]	m	}					Н	Э	н	э
E			.	.	N		n						О	Ю	о	ю
F			/	?	O	_	o						П	Я	п	я

2 ЛАБОРАТОРНА РОБОТА №1 АРХІТЕКТУРА ЕОМ І СИСТЕМА КОМАНД

Мета роботи

Знайомство з інтерфейсом моделі ЕОМ, методами введення і налагодження програми, діями основних класів команд і способів адресації.

2.1 Теоретичні відомості

Для вирішення за допомогою ЕОМ деякої задачі повинна бути розроблена програма. Програма на мові ЕОМ є послідовністю команд. Код кожної команди визначає виконувану операцію, тип адресації і адресу. Виконання програми, записаної в пам'яті ЕОМ, здійснюватись послідовно по командам в порядку зростання, адрес команд або в порядку, який визначається командами передачі управління.

Для того щоб отримати результат виконання програми, користувач повинен:

– ввести програму в пам'ять ЕОМ;

– визначити, якщо це необхідно, вміст комірок ОЗП і РОН, що містять вихідні дані, а також регістрів IR і BR;

– встановити в PC стартовий адресу програми;

– перевести модель в режим Робота.

Кожне з цих дій виконується за допомогою інтерфейсу моделі, описаного в теоретичних відомостях. Введення програми може здійснюватися як в машинних кодах безпосередньо в пам'ять моделі, так і в мнемокодах в вікно Текст програми з подальшим асемблюванням.

Мета цієї лабораторної роботи - знайомство з інтерфейсом моделі ЕОМ, методами введення і налагодження програми, діями основних класів команд і способів адресації. Для цього необхідно ввести в пам'ять ЕОМ і виконати в режимі Крок деяку послідовність команд (певну варіантом завдання) і зафіксувати всі зміни на рівні програмно-доступних об'єктів ЕОМ, що відбуваються при виконанні цих команд.

Команди в пам'ять навчальної ЕОМ вводяться в вигляді шестирозрядних десяткових чисел.

У цій лабораторній роботі програмуватимемо ЕОМ в машинних кодах.

2.2 Приклад виконання завдання

Дана послідовність Мнемо-кодів, яку необхідно перетворити в машинні коди, занести в ОЗП ЕОМ, виконати в режимі Крок і зафіксувати зміну станів програмно-доступних об'єктів ЕОМ (Таблиця 2.3).

Таблиця 2.1 – Команди та коди

Послідовність	**Значення**				
Команди	RD # 20	WR 30	ADD # 5	WR @ 30	JNS 02
Коди	21 1 020	22 0 030	23 1 05	22 2 30	12 0 002

Введемо отримані коди послідовно в комірки ОЗП, починаючи з адреси 000. Виконуючи команди в режимі Крок, будемо фіксувати зміни програмно-доступних об'єктів (в даному випадку це Acc, PC і комірки ОЗП 020 і 030) (Таблиця 2.2).

Таблиця 2.2 – Вміст регістрів

PC	**Acc**	**M(30**	**M(20)**
000	000000	000000	
001	000020		
002		000020	
003	000025		
004			000025
002			
003		000035	
004			000030

2.3 Завдання на лабораторну роботу

1. Ознайомитися з архітектурою ЕОМ (див. Теорію).
2. Записати в ОЗП "програму", що складається з команд відповідно до варіанту з таблиці 1.3. Команди розмістити в послідовних комірках пам'яті.
3. При необхідності встановити початкове значення IR.
4. Визначити ті програмно-доступні об'єкти ЕОМ, які будуть змінюватися при виконанні цих команд.
5. Виконати в режимі Крок введену послідовність команд, фіксуючи зміни значень об'єктів, визначених у пункті 4.
6. Якщо в програмі утворюється цикл, необхідно переглянути не більше двох повторень кожної команди, що входить в тіло циклу.

Таблиця 2.3 – Варіанти завдання

№	IR	Команда 1	Команда 2	Команда 3	Команда 4	Команда 5
1	2	3	4	5	6	7
1	000007	IN	MUL #2	WR 10	WR @10	JNS 001
2	X	RD #17	SUB #9	WR 16	WR @16	JNS 001
3	100029	IN	ADD #16	WR 8	WR @8	JS 001
4	X	RD #2	MUL #6	WR 11	WR @11	JNZ 00
5	000016	IN	WR 8	DIV @4	WR @8	JMP 002
6	X	RD #4	WR 11	RD @11	ADD #330	JS 000
7	000000	IN	WR 9	RD @9	SUB #1	JS 001
8	X	RD 4	SUB #8	WR 8	WR @8	JNZ 001
9	100005	IN	ADD #12	WR 10	WR @10	JS 004
10	X	RD 4	ADD #15	WR 13	WR @13	JMP 001
11	000315	IN	SUB #308	WR 11	WR @11	JMP 001
12	X	RD #988	ADD #19	WR 9	WR @9	JNZ 002
13	000017	IN	WR 11	ADD 11	WR @11	JMP 002
14	X	RD #5	MUL #9	WR 10	WR @10	JNZ 001

2.4 Зміст звіту

1. Формулювання варіанта завдання;
2. Машинні коди команд, відповідних варіанту завдання;
3. Результати виконання послідовності команд в формі таблиці.

2.5 Контрольні питання

1. З яких основних частин складається ЕОМ, і які з них представлені в моделі?
2. Що таке система команд ЕОМ?
3. Які класи команд представлені в моделі?
4. Які дії виконують команди передачі управління?
5. Які способи адресації використані в моделі ЕОМ? У чому відмінність між ними?
6. Які обмеження накладаються на спосіб представлення даних в моделі ЕОМ?
7. Які режими роботи передбачені в моделі і в чому відмінність між ними?
8. Як записати програму в машинних кодах в пам'ять моделі ЕОМ?
9. Як переглянути вміст регістрів процесора і змінити вміст деяких регістрів?
10. Як переглянути і, при необхідності, відредагувати вміст комірки пам'яті? Як запустити виконання програми в режимі призупинення роботи після виконання кожної команди?
11. Які способи адресації операндів застосовуються в командах ЕОМ?
12. Які команди належать до класу передачі управління?

3 ЛАБОРАТОРНА РОБОТА №2 РОЗГАЛУЖЕННЯ У ПРОГРАМІ І ПЕРЕАДРЕСАЦІЯ

Мета роботи

Практична реалізація алгоритмів, шляхи в яких залежать від вихідних даних, з використанням команд умовної передачі управління. Рішення задач, пов'язаних з обробкою масивів, застосуванням спеціальних видів адресації.

3.1 Програмування розгалуженого процесу

3.1.1 Приклад

В якості прикладу (кілька спрощеного в порівнянні з завданнями лабораторної роботи № 2) розглянемо програму обчислення функції, при чому X вводиться з пристрою введення IR, результат виводиться на OR.

$$y = \begin{cases} (x-11)^2 - 125, & \text{при } x \geq 16, \\ \dfrac{x^2 + 72x - 6400}{-168}, & \text{при } x < 16, \end{cases}$$

Схема алгоритму розв'язання задачі показана на рисунку 3.1

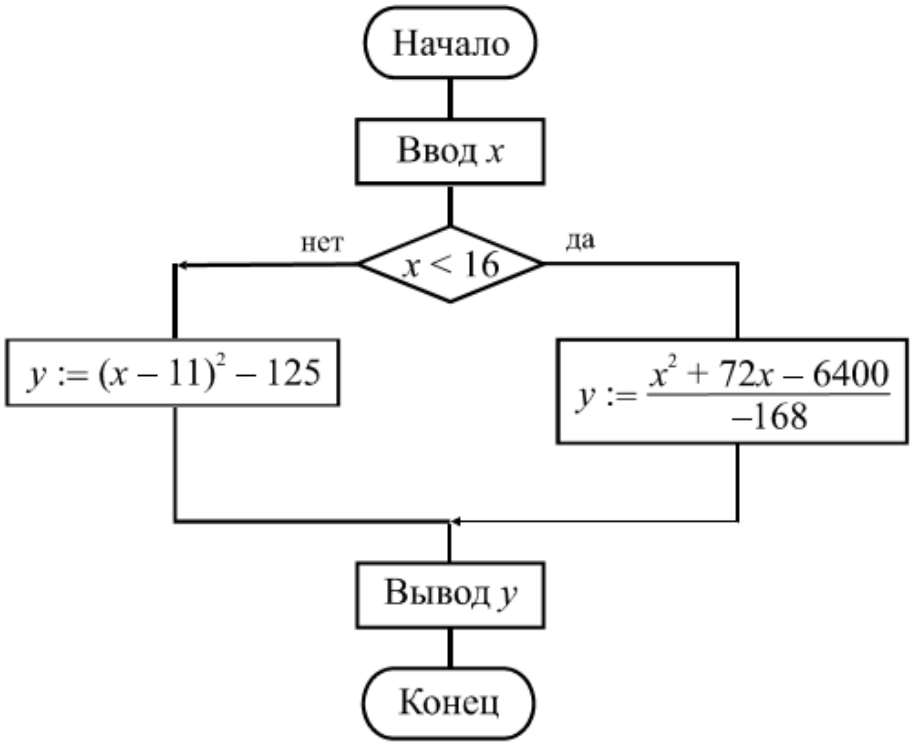

Рисунок 3.1 – Граф-схема алгоритму

У даній лабораторній роботі використовуються дві умовні команди з безпосередньою адресацією, що дозволяють оперувати від'ємними числами і числами по модулю, що перевищують 999, в якості безпосереднього операнда.

Оцінивши розмір програми приблизно в 20 - 25 команд, відведемо для області даних комірки ОЗП, починаючи з адреси 030. Складена програма з коментарями представлена у вигляді таблиці (таблиця 3.1).

Таблиця 3.1 – Приклад програми

Адр.	Команда		Коментар
	Мнемокод	Код	
000	IN	01 0 000	Введення x
001	WR 30	22 0 030	Розміщення x в ОЗП
002	SUB #16	24 1 016	Порівняння з межею (x-16)
003	JS 010	13 0 010	Перехід по негативного результату
004	RD 30	21 0 030	Обчислення за формулою
005	SUB #11	24 1 011	
006	WR 31	22 0 031	
007	MUL 31	25 1 125	
008	SUB #125	24 1 125	
009	JMP 020	10 0 020	Перехід на вивід результату
010	RD 30	21 0 030	Обчислення по другій формулі
011	MUL 30	25 0 030	
012	WR 31	22 0 031	
013	RD 30	21 0 030	
014	MUL #72	25 1 072	
015	ADD 31	23 0 031	
016	ADI 106400	43 0 000	
017		106400	
018	DIVI 100168	16 0 000	
019		100168	
020	OUT	02 0 000	Вивід результату
021	HTL	09 0 000	Стоп

3.1.2 Завдання на лабораторну роботу

1. Розробити програму обчислення і виведення значення функції для введеного з IR значення аргументу x. Функції та допустимі межі зміни аргументу наведені в таблиці 3.2, варіанти завдань в таблиці 3.3.

$$y = \begin{cases} F_i(x), & \text{при } x \geq a, \\ F_j(x), & \text{при } x < a, \end{cases}$$

2. Виходячи з допустимих меж зміни аргументу функцій (таблиця 3.2) і значення параметра a для свого варіанту завдання (таблиця 3.3) виділити на числовій осі Ox області, в яких функція y обчислюється за представленою в п. 1 формулою, і неприпустимі значення аргументу.

На неприпустимих значеннях аргументу програма повинна видавати на OR максимальне негативне число: 199 999.

3. Ввести текст програми в вікно Текст програми, при цьому можливий набір і редагування тексту безпосередньо у вікні Текст програми або завантаження тексту з файлу, підготовленого в іншому редакторі.

4. Асемблювати текст програми, при необхідності виправити синтаксичні помилки.

5. Налагодити програму. Для цього:

а) записати в IR значення аргументу x> a (в області допустимих значень);

б) Записати в PC стартовий адресу програми;

в) Перевірити правильність виконання програми (т. е. Правильність результату і адреси зупинки) в автоматичному режимі. У разі наявності помилки виконати пп. 5, г і 5, д; інакше перейти до п. 5, е;

г) Записати в PC стартову адресу програми;

д) Спостерігаючи виконання програми в режимі Крок, знайти команду, яка є причиною помилки; виправити її; виконати пп. 5, а - 5, в;

е) Записати в IR значення аргументу x <a (в області допустимих значень); виконати пп. 5, б і 5, в;

ж) Записати в IR неприпустиме значення аргументу x і виконати пп. 5, б і 5, ст.

6. Для обраного допустимого значення аргументу x спостерігати виконання налагодженої програми в режимі Крок і записати у формі таблиці вміст регістрів ЕОМ перед виконанням кожної команди.

Таблиця 3.2 – Функції

k	$F_k(x)$	k	$F_k(x)$
1	$\frac{x+17}{1-x};\ 2 \le x \le 12$	5	$\frac{(x+2)^2}{15};\ 50 \le x \le 75$
2	$\frac{(x+3)^2}{x};\ 1 \le x \le 50$	6	$\frac{2x^2+7}{x};\ 1 \le x \le 30$
3	$\frac{1000}{x+10};\ -50 \le x \le -15$	7	$\frac{x^2+2x}{10};\ -50 \le x \le 50$
4	$(x+3)^3;\ -20 \le x \le 20$	8	$\frac{8100}{x^2};\ 1 \le x \le 90$

Таблиця 3.3 – Варіанти завдань

№ варіанту	i	j	a	№ варіанту	i	j	a
1	2	1	12	8	8	6	30
2	4	3	-20	9	2	6	25
3	8	4	15	10	5	7	50
4	6	1	12	11	2	4	18
5	5	2	50	12	8	1	12
6	7	3	15	13	7	6	25
7	6	2	11	14	1	4	5

3.2 Зміст звіту

1. Формулювання варіанта завдання.
2. Граф-схема алгоритму розв'язання задачі.
3. Розміщення даних в ОЗП.
4. Програма в формі таблиці.
5. Послідовність станів регістрів ЕОМ при виконанні програми в режимі Крок для одного значення аргументу.
6. Результати виконання програми для декількох значень аргументу, обраних самостійно.

3.3 Контрольні питання

1. Як працює механізм непрямої адресації?
2. Яка комірка буде адресована в команді з непрямою адресацією через комірку 043, якщо вміст цієї комірки рівний 102 347?
3. Як працюють команди передачі управління?
4. Що входить в поняття "налагодження програми"?
5. Які способи налагодження програми можна реалізувати в моделі?

3.4 Програмування циклу з переадресацією

3.4.1 Приклад

Розробити програму обчислення суми елементів масиву чисел Су, С2, .., Cn. Вихідними даними в цьому завданні є: n - кількість сумованих чисел і Су, С2, ..., Сп - масив сумованих чисел. Зауважимо, що повинна виконуватися умова n> 1, т. К. Алгоритм передбачає, принаймні, одне підсумовування. Крім того, передбачається, що підсумовувані числа записані в ОЗП поспіль, т. е. В комірці пам'яті з послідовними адресами. Результатом є сума S.

Складемо програму для обчислення суми з наступними конкретними параметрами: число елементів масиву - 10, елементи масиву розташовані в комірках ОЗП за адресами 040, 041, 042, ..., 049. Використовувані для рішення задачі проміжні змінні мають наступний сенс : Ai - адреса числа Ci, i є {l, 2, ..., 10}; ОЗП (Ai) - число за адресою Ai, S - поточна сума; k - лічильник циклу, що визначає число повторень тіла циклу.

Розподіл пам'яті такий. Програму розмістимо в комірках ОЗП, починаючи з адреси 000, приблизна оцінка обсягу програми - 20 команд; проміжні змінні: Ai - в комірці ОЗП з адресою 030, k - за адресою 031, S - за адресою 032. Схема алгоритму програми показана на малюнку 3.2, текст програми з коментарями наведено в таблиці 3.4.

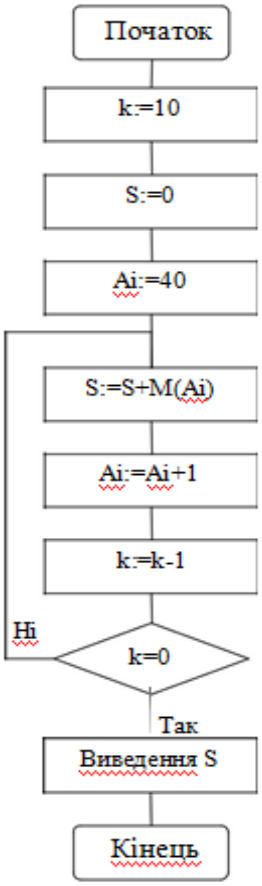

Рисунок 3.2 – Схема алгоритму для прикладу

Таблиця 3.4 – Текст програми прикладу

Адр.	**команда**	**Примітка**
000	RD # 40	Завантаження початкового адресу масиву 040
001	WR 30	в комірку 030
002	RD # 10	Завантаження параметра циклу k = 10 в комірку 031
003	WR 31	
004	RD # 0	Завантаження початкового значення суми S = 0
005	WR 32	в комірку 032
006	M1: RD 32	Додавання
007	ADD @ 30	до поточної суми
008	WR 32	чергового елемента масиву
009	RD 30	модифікація поточного
010	ADD # 1	адресу масиву
011	WR 30	(Перехід до наступного адресу)

012	RD 31	зменшення лічильника
013	SUB # 1	(Параметра циклу)
014	WR 31	на 1
015	JNS M1	Перевірка параметра циклу і перехід при k ≠ 0
016	RD 32	Виведення
017	OUT	результату
018	HTL	стоп

3.4.2 Завдання на лабораторну роботу

1. Написати програму визначення заданої характеристики послідовності чисел С1, С2, .., Сп. Варіанти завдань наведені в таблиці 3.5.

2. Записати програму в мнемокодах, ввівши її в поле вікна Текст програми.

3. Зберегти набрану програму у вигляді текстового файлу і виконати ассемблерування Мнемокоду.

4. Завантажити в ОЗП необхідні константи і вихідні дані.

5. Налагодити програму

Таблиця 3.5 – Завдання по варіантах

№ варіанту	**Характеристики послідовності С**
1	Кількість парних чисел
2	Номер мінімального числа
3	Добуток всіх чисел
4	Номер першого негативного числа
5	Кількість чисел рівних C_1
6	Кількість негативних чисел
7	Максимальне від'ємне число
8	Номер першого позитивного числа
9	Мінімальне позитивне число
10	Номер максимального числа
11	Кількість непарних чисел
12	Кількість чисел менших C_1
13	Різниця між сумою парних і непарних елементів масиву
14	Відношення сум парних і непарних елементів масиву

3.5 Зміст звіту

1. Формулювання варіанта завдання.
2. Граф-схема алгоритму розв'язання задачі.
3. Розподіл пам'яті (розміщення в ОЗП змінних, програми та необхідних констант).

4. Програма.
5. Значення вихідних даних і результату виконання програми.

3.6 Контрольні питання

1. Як організувати цикл в програмі?
2. Що таке параметр циклу?
3. Як поведе себе програма, наведена в табл. 2. , якщо в ній буде відсутня команда wr 31 за адресою 014?
4. Як поведе себе програма, наведена в табл. 2. , якщо мітка m1 буде поставлена за адресою 005? 007?

4 ЛАБОРАТОРНА РОБОТА № 3 ПІДПРОГРАМИ І СТЕК. КОМАНДНИЙ ЦИКЛ ПРОЦЕСОРА

Мета роботи

Реалізація підпрограм. Дослідження роботи мікрокоманд.

4.1 Реалізація підпрограм и використання стека

У програмуванні часто зустрічаються ситуації, коли однакові дії необхідно виконувати багаторазово в різних частинах програми (наприклад, обчислення функції sinx). При цьому з метою економії пам'яті не слід багаторазово повторювати одну і ту ж послідовність команд - достатньо один раз написати так звану підпрограму (в термінах мов високого рівня - процедуру) і забезпечити правильний виклик підпрограми та повернення в точку виклику по завершенню підпрограми.

Для виклику підпрограми необхідно вказати її початковий адресу в пам'яті і передати (якщо необхідно) параметри - ті вихідні дані, з якими будуть виконуватися передбачені в підпрограмі дії. Адреса підпрограми вказується в команді виклику call, а параметри можуть передаватися через певні комірки пам'яті, регістри або стек.

Повернення в точку виклику забезпечується збереженням адреси поточної команди (вмісту регістра PC) при виклику і використанням в кінці підпрограми команди повернення ret, яка повертає збережене значення адреси повернення в PC.

Для реалізації механізму вкладених підпрограм (можливість виклику підпрограми з іншої підпрограми тощо) адреси повернення доцільно зберігати в стеку. Стек ("магазин") - особливим чином організована безадресна пам'ять, доступ до якої здійснюється через одну комірку, звану верхівкою стека. При записі слово поміщається у верхівку стека, попередньо всі наявні в ньому слова зміщуються вниз на одну позицію; при читанні витягується вміст верхівки стека (воно при цьому з стека зникає), а всі слова зміщуються вгору на одну позицію. Такий механізм нагадує дію магазину стрілецької зброї (звідси і друга назва). У програмуванні називають таку дисципліну обслуговування LIFO (Last In First Out, останнім прийшов - першим вийшов) на відміну від дисципліни типу чергу - FIFO (First In First Out, первим прийшов - першим вийшов).

У звичайних ОЗП немає можливості переміщувати слова між комірками, тому при організації стека переміщається не масив слів відносно нерухомої верхівки, а верхівка щодо нерухомого масиву. Під стек відводиться певна область ОЗП, причому адресу верхівки зберігається в спеціальному регістрі процесора - покажчику стека SP.

У стек можна помістити вміст регістра загального призначення по команді push або витягти вміст верхівки в регістр загального призначення по команді pop. Крім того, по команді виклику підпрограми call значення програмного лічильника PC (адреса наступної команди) поміщається в верхівку стека, а по команді ret вміст верхівки стека витягується в PC. При кожному зверненні в стек покажчик SP автоматично модифікується.

У більшості ЕОМ стек росте в бік менших адрес, тому перед кожним записом вміст SP зменшується на 1, а після кожного запису вміст SP збільшується на 1. Таким чином, SP завжди вказує на верхівку стека.

Мета цієї частини лабораторної роботи - вивчення організації програм з використанням підпрограм. Крім того, в процесі організації циклів ми будемо використовувати нові можливості системи команд моделі ЕОМ, які дозволяють працювати з новим класом пам'яті - сверхоперативної (регістри загального призначення - РОН). У реальних ЕОМ доступ в РОН займає значно менше часу, ніж в ОЗП; крім того, команди поводження з регістрами коротше команд звернення до пам'яті. Тому в РОН розміщуються найбільш часто використовувані в програмі дані, проміжні результати, лічильники циклів, непрямі адреси і т. д.

У системі команд навчальної ЕОМ для роботи з РОН використовуються спеціальні команди, мнемоніки яких збігаються з мнемоніками відповідних команд для роботи з ОЗП, але в адресній частині містять символи регістрів R0-R9.

Крім звичайних способів адресації (прямої та непрямої) в регістрових команди використовуються два нових - постінкрементна і преднінкрементна. Крім того, до регістрових відноситься команда організації циклу JRNZ R,M. По цій команді вміст зазначеного в команді регістра зменшується на 1, і якщо в результаті віднімання вмісту регістра не дорівнює 0, то керування передається на мітку м. Цю команду слід ставити в кінець тіла циклу, мітку м - в першій команді тіла циклу, а в регістр R поміщати число повторень циклу.

4.1.1 Приклад

Дано три масива чисел. Потрібно обчислити середнє арифметичне їх максимальних елементів. Кожен масив задається двома параметрами: адресом першого елемента і довжиною. Очевидно, у програмі необхідно тричі виконати пошук максимального елемента масиву, тому слід написати відповідну підпрограму.

Параметри в підпрограму будемо передавати через регістри: R1 - початкова адреса масиву, R2 - довжина масиву.

Розглянемо конкретну реалізацію цієї задачі. Нехай перший массив починається з адреси 085 і має довжину 14 елементів, другий - 100 і 4, третій - 110 і 9. Програма буде складатися з основної частини та підпрограми. Основна програма задає параметри підпрограмі, що викликає її і зберігає результати роботи підпрограми в робочих клітинок. Потім здійснює обчислення середнього арифметичного і виводить результат на пристрій виведення. В якості робочих клітинок використовуються регістри загального призначення R6 та R7 - для зберігання максимальних елементів масивів. Підпрограма отримує параметри через регістри R1 (початкова адреса масиву) і R2 (довжина масиву).

Ці регістри використовуються підпрограмою в якості поточного регістра адреси і лічильника циклу відповідно. Крім того, R3 використовується для

збереження поточного максимуму, а R4 - для тимчасового збереження поточного елемента.

Підпрограма повертає результат через акумулятор. В таблиці 3.1 наведено текст основної програми та підпрограми. Зверніть увагу, цикл в підпрограмі організований за допомогою команди jrnz, а модифікація поточного адресу - засобами постінкрементної адресації.

Таблиця 4.1 – Програма прикладу

Команда	Примітка
Основна програма	
RD #85	Завантаження
WR R1	параметрів
RD #14	першого
WR R2	масиву
CALL M	Виклик підпрограми
WR R6	Збереження результату
RD #100	Завантаження
WR R1	параметрів
RD #4	другого
WR R2	масиву
CALL M	Виклик підпрограми
WR R7	Збереження результату
RD #110	Завантаження
WR R1	параметрів
RD #9	третього
WR R2	масиву
CALL M	Виклик підпрограми
ADD R7	Обчислення
ADD R6	середнього
DIV #3	арифметичного
OUT	Виклик результату
HLT	Стоп
Підпрограма MAX	
M: RD @R1	Завантаження
WR R3	першого элементу в R3
L2: RD @R1+	Зчитування элементу і модифікація адреси
WR R4	Порівняння
SUB R3	і заміна
JS L1	Якщо R3<R4
MOV R3, R4	
L1: JRNZ R2, L2	Цикл
RD R3	Зчитування результату в Acc
RET	Повернення

4.1.2 Завдання

Скласти та налагодити програму навчальної ЕОМ для вирішення наступного завдання . Три масива в пам'яті задані початковими адресами і довжинами. Обчислити та вивести на пристрій виводу середнє арифметичне параметрів цих масивів. Параметри визначаються завданням до попередньої лабораторної роботи (таблиця 4.5), причому відповідність між номерами варіантів завдань 2 і 3 встановлюється по таблиці 4.2

Таблиця 4.2 – Відповідність між номерами завдань

Номер варіанту	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Номер рядка в табл. 4.5	5	7	13	11	9	12	1	10	14	3	6	8	2	4

4.1.3 Зміст звіту

1. Формулювання варіанту завдання.
2. Схема алгоритму основної програми.
3. Схема алгоритму підпрограми.
4. Розподіл пам'яті (розміщення в ОЗП змінних, програми і необхідних констант).
5. Тексти програми і підпрограми.
6. Значення вихідних даних та результату виконання програми.

4.1.4 Контрольні питання

1. Як працює команда mov R3, R7?
2. Які дії виконує процесор при реалізації команди call?
3. Як поведе себе програма прикладу 4, якщо в ній замість команд call m використовувати команди jmp m?
4. Після початкової установки процесора (сигнал Скидання) покажчик стека SP встановлюється 000. За якою адресою буде проводитися запис в стек перший раз, якщо не завантажувати SP командою wrsp?
5. Як, використовуючи механізми постинкрементної і преінкрементної адресації, організувати додатковий стек в довільній області пам'яті, не пов'язаний з SP?

4.2 Дослідження командного циклу процесора

Реалізація програми ЕОМ зводиться до послідовного виконання команд. Кожна команда, у свою чергу, виконується як послідовність мікрокоманд, що реалізують елементарні дії над операційними елементами процесора.

У програмній моделі навчальної ЕОМ передбачений Режим мікрокоманд, в якому дія командного циклу реалізується і відображається на рівні мікрокоманд. Список мікрокоманд поточної команди відображається в спеціальному вікні Мікрокоманди.

4.2.1 Завдання 1

Виконати знову послідовність команд за варіантом завдання 1 (таблиця 2.3 з 1 лабораторної роботи), але в режимі Шаг. Зареєструвати зміни стану процесора і пам'яті у формі таблиці 4.3, в якій наведено стани ЕОМ при виконанні прикладу (фрагменту).

4.2.2 Завдання 2

Записати послідовність мікрокоманд для наступних команд моделі навчальної ЕОМ:

ADD R3
ADD @R3
ADD @R3+
ADD -@R3
JRNZ R3,M
MOV R4,R2
JMP M
CALL M
RET: PUSH R3
POP R5

4.2.3 Контрольні питання

1. Які мікрокоманди пов'язані зі зміною стану акумулятора?
2. Які дії виконуються в моделі за мікрокомандою MRd? RWr?
3. Спробуйте скласти мікропрограму (послідовність мікрокоманд, що реалізують команду) для неіснуючої команди "множення модулів чисел".
4. Що зміниться в роботі процесора, якщо в кожній мікропрограмі микрокоманду збільшення програмного лічильника PC := PC + 1 перемістити в кінець мікропрограми?

Таблиця 4.3 – Стан моделі в режимі моделювання на рівні мікрокоманд

Комірки		АУ		CR			ОЗП		Мікрокоманда	Мнемокод	Адрес(PC)
30	20	DR	Acc	ADR	TA	COP	MDR	MAR			
0	0	0	0	0	0	0	0	0	MAR:=PC	RD #20	0
								0	MRd		
							211020		CR:=MDR		
				20	1	21			PC:=PC+1		
									Acc:=000.ADR		1
			20						MAR:=PC	WR 30	
								1	MRd		
							220030		CR:=MDR		
				30	0	22			PC:=PC+1		
									MAR:=ADR		2
								30	MDR:=Acc		
							20		MWr		
20									MAR:=PC	ADD #5	
								2	MRd		
							231005		CR:=MRD		
				5	1	23			PC:=PC+1		
									DR:=000.ADR		3
		5							F_{AY}:=ALI		
			25						MAR:=PC	WR @30	

5 ЛАБОРАТОРНА РОБОТА № 4. ПРОГРАМУВАННЯ ЗОВНІШНІХ ПРИСТРОЇВ

Мета роботи

Вивчення способів організації взаємодії процесора і зовнішніх пристроїв (ЗП) у складі ЕОМ.

Вище зазначалося, що зв'язок процесора і ЗП може здійснюватися в синхронному або асинхронному режимі. Синхронний режим використовується для ЗП, завжди готових до обміну. В нашій моделі такими ЗП є дисплей і тоногенератор - процесор може звертатися до цих ЗП, не аналізуючи їх стан (правда дисплей блокує прийом даних після вводу 128 символів, формуючи прапор помилки).

Асинхронний обмін передбачає аналіз процесором стану ЗП, який визначає готовність ЗП видати або прийняти дані або факт здійснення деякої події, контрольованого системою. До таких пристроїв в нашій моделі можна віднести клавіатуру і блок таймерів.

Аналіз стану ЗП може здійснюватися процесором двома способами:

– програмно-керованому режимі;

– в режимі переривання.

У першому випадку передбачається програмне звернення процесора до регістру стану ЗП з подальшим аналізом значення відповідного розряду слова стану. Таке звернення слід передбачити в програмі з певною періодичністю, незалежно від фактичного настання контрольованого події (наприклад, натискання клавіші).

У другому випадку при виникненні контрольованої події ЗП формує процесору запит на переривання програми, по якій процесор і здійснює зв'язок з ЗП.

5.1 Завдання

Свій варіант завдання (таблиці 5.1) потрібно виконати двома способами - спочатку в режимі програмного контролю, далі модифікувати програму таким чином, щоб події оброблялися в режимі переривання програми. Оскільки "фонова" (основна) завдання для цього випадку в завданнях відсутня, її роль може зіграти "порожній цикл":

M: *NOP*
NOP
JMP ***M***

Таблиця 5.1 – Варіанти завдання

№ варіанта	Завдання (Використовувані ЗП)	Пояснення
1	Введення пятиразрядных чисел комірки ОЗУ. (Клавіатура)	Програма повинна забезпечувати введення послідовності символів ASCII - кодів десяткових цифр не довше п'яти, перекодування в «8421», упаковку в десяткове число (перший введений символ – старша цифра) і розміщення в комірці ОЗП. ASCII- коди не-цифр ігнорувати.
2	Програма введення символів з клавіатури з виведенням на дисплей. (Клавіатура, дисплей, таймер)	Очищення буфера клавіатури після введення 50 символів або кожні 10 с.
3	Виведення на дисплей трьох текстів, зберігаються в пам'яті, із затримкою. (Дисплей, таймер)	Перший текст виводиться відразу при запуску програми, другий через 15с, третій – через 20с після другого.
4	Виведення на дисплей одного з трьох текстових повідомлень. (Клавіатура, дисплей)	<1> – виведення на дисплей першого текстового повідомлення, <2> – другого, <3> – третього, інші символи – немає реакції.
5	Вибрати з потоку ASCII-кодів тільки цифри і виводити їх на екран. (Клавіатура, дисплей, тоногенератор)	Виведення кожної цифри супроводжується коротким звуковим сигналом.
6	Вводити на дисплей кожен введений символ, цифру вводити «у трьох екземплярах». (Клавіатура, дисплей)	Виведення кожної цифри супроводжується трьохразовим звуковим сигналом.
7	Селективне введення символів з клавіатури. (Клавіатура, дисплей)	Всі російські літери, що зустрічаються в рядку введення – у верхню частину екрана дисплея (рядки 1-4), всі цифри – в нижню частину екрану (рядки 5-8), інші символи не виводити.
8	Виведення заданого вмісту ділянки пам'яті на дисплей символи з заданим проміжком часу між виведенням символів. (Дисплей, таймер)	Залишок від ділення на 256 трьох молодших розрядів комірки пам'яті розглядається як ASCII-код символу. Початковий адрес пам'яті, довжина масиву виводу та проміжок часу – параметри підпрограми.
9	Програма введення символу з клавіатури з виведенням на дисплей. (Клавіатура, дисплей)	Очищення буфера клавіатури після введення 35 символів.

5.2 Завдання для захисту

1. Розробити програму-тест на швидкість введення символів з клавіатури. По звуковому сигналу включається клавіатура і таймер на T секунд. Можна починати введення символів, причому кожен символ відображається на дисплеї, ведеться підрахунок кількості введених символів (після кожних 50 дається команда на очищення буфера клавіатури, після 128 - очищається дисплей). Переповнення таймера вимикає клавіатуру і включає сигнал завершення введення (можна тон цього сигналу співставити з кількістю введених символів). Параметр T вводиться з IR. Результат S - середня швидкість введення (символ/с) видається на OR. Враховуючи, що модель навчальної ЕОМ оперує тільки цілими числами, можна видавати результат у форматі 5x60 символів/хв.
2. Розробити програму-тест на ступінь запам'ятовування тексту. Три різних варіанта тексту виводяться послідовно на дисплей на T1 секунд з проміжками T2 секунд. Далі ці тексти (те, що запам'яталося) вводяться з клавіатури (в режимі введення рядка) і программно порівнюються з вихідними текстами. Видається кількість (відсоток) помилок.
3. Розробити програму-калькулятор. Здійснювати введення з буфера клавіатури послідовності цифр, упаковку (див. завдання 1 в табл. 2.12).
4. Роздільники - знаки арифметичних бінарних операцій и =. Результат переводиться в ASCII-коди і виводиться на дисплей.

5.3 Порядок виконання роботи

1. Запустити програмну модель навчальної ЕОМ і підключити до неї визначені у завданні зовнішні пристрої (меню Зовнішні пристрої | Менеджер ЗП).
2. Написати і відлагодити програму, передбачену завданням, з використанням програмного аналізу прапорів готовності ЗП. Продемонструвати працюючу програму вчителю.
3. Змінити налагоджену в п. 2 програму таким чином, щоб процесор реагував на готовність ЗП за допомогою підсистеми преривання. Продемонструвати роботу зміненої програми вчителю.

5.4 Зміст звіту

1. Текст програми з програмним аналізом прапорів готовності ЗП.
2. Текст програми з обробником преривання.

5.5 Контрольні питання

1. При яких умовах встановлюється і скидається прапор готовності клавіатури Rd?
2. Можливо в блоці таймерів організувати роботу всіх трьох таймерів з різною тактовою частотою?

3. При отриманні запиту на переривання від блоку таймерів визначити номер таймера, що досяг стану 99 999 (00 000)?

4. Який текст виявиться на екрані дисплея, якщо після натискання у вікні оглядача дисплея кнопки Очистити і завантаження за адресою CR (11) константи #10 вивести за адресою DR (10) послідовно п'ять ASCII-кодів російських букв А, Б, В, Г, Д?

5. В якій області пам'яті моделі ЕОМ можуть розташовуватися програми - обробники переривань?

6. Які зміни в роботі налагодженої вами другої програми відбудуться, якщо завершити обробник переривань командою ret, а не iret?

6 ЛАБОРАТОРНА РОБОТА № 5. ПРИНЦИПИ РОБОТИ КЕШ-ПАМ'ЯТІ

Мета роботи

Перевірити роботу різних алгоритмів заміщення при різних режимах запису.

6.1 Завдання 1

В якості завдання пропонується деяка коротка програма (таблиця 6.2), яку необхідно виконати з підключеною кеш-пам'яттю (розміром 4 і 8 комірок) в пошаговому режимі для таких двох варіантів алгоритмів заміщення (6.1) див. так само теоретичні відомості.

Таблиця 6.1 – Пояснення до варіантів завдання

Номер варіанта	Режим запису	Алгоритм заміщення
1,7,11	Наскрізна	СЗ, без урахування біта запису
	Зворотня	О, з урахування біта запису
2,5,9	Наскрізна	БИ, без урахування біта запису
	Зворотня	О, з урахування біта запису
3,6,12	Наскрізна	О, без урахування біта запису
	Зворотня	СЗ, з урахування біта запису
4,8,10	Наскрізна	БИ, без урахування біта запису
	Зворотня	БИ, з урахування біта запису

Таблиця 5.2 – Варіанти завдань

№ вар.	Номера команд программы						
	1	2	3	4	5	6	7
1	RD #12	WR 10	WR @10	ADD 12	WR R0	SUB 10	PUSH R0
2	RD #65	WR R2	MOV R4,R2	WR 14	PUSH R2	POP R3	CALL 002
3	RD #16	SUB #5	WR 9	WR @9	WR R3	PUSH R3	POP R4
4	RD #99	WR R6	MOV R7,R6	ADD R7	PUSH R7	CALL 006	POP R8
5	RD #11	WR R2	WR -@R2	PUSH R2	CALL 005	POP R3	RET
6	RD #19	SUB #10	WR 9	ADD #3	WR @9	CALL 006	POP R4
7	RD #6	CALL 006	WR 11	WR R2	PUSH R2	RET	JMP 002
8	RD #8	WR R2	WR @R@+	PUSH R2	POP R3	WR -@R3	CALL 003
9	RD #13	WR 14	WR @14	WR @13	ADD 13	CALL 006	RET
10	RD #42	SUB #54	WR 16	WR @16	WR R1	ADD @R1	PUSH R1
11	RD #10	WR R5	ADD R5	WR R6	CALL 005	PUSH R6	RET
12	JMP 006	RD #76	WR 14	WR R2	PUSH R2	RET	CALL 001

Не слід розглядати задану послідовність команд як фрагмент програми. Деякі конструкції, наприклад, послідовність команд push R6, ret в загальному випадку не повертає програму в точку виклику підпрограми. Такі групи команд

введені в завдання для того, щоб звернути увагу на особливості функціонування стека.

6.1.1 Порядок виконання роботи

Ввести в модель навчальної ЕОМ текст свого варіанту програми (таблиця 6.2), асемблювати його і зберегти на диску у вигляді txt-файл.

Встановити параметри кеш-пам'яті розміром 4 комірки, вибрати режим запис та алгоритм заміщення згідно з першим рядком свого варіанту з таблиці 6.1.

В пошаговому режимі виконати програму, фіксуючи після кожного кроку стан кеш-пам'яті.

Для однієї з команд запису (WR) перейти в режим Такт та зазначити, в яких мікрокомандах відбувається зміна кеш-пам'яті.

Для кеш-пам'яті розміром 8 комірок встановити параметри відповідно з другим рядком свого варіанту з таблиці 6.1 і виконати програму в по-шаговому режимі ще раз, фіксуючи послідовність номерів заміщаючих комірок кеш-пам'яті.

6.1.2 Зміст звіту

1. Варіант завдання — текст програми та режими кеш-пам'яті.
2. Послідовність станів кеш-пам'яті розміром 4 комірки при одноразовому виконанні програми (команди 1-7).
3. Послідовність мікрокоманд при виконанні команди wr з відміткою тих мікрокоманд, в яких можлива модифікація кеш-пам'ять.
4. Для варіанта кеш-пам'яті розміром 8 комірок – послідовність номерів заміщуються комірки кеш-пам'яті для другого варіанту параметрів кеш пам'яті при дворазовому виконанні програми (команди 1-7).

6.1.3 Контрольні питання

5. У чому сенс включення кеш-пам'яті до складу ЕОМ?
6. Як працює кеш-пам'ять в режимі зворотного запису? Наскрізний запису?
7. Як залежить ефективність роботи ЕОМ від розміру кеш-пам'яті?
8. В яку комірку кеш-пам'яті буде поміщатися чергове слово, якщо вільні комірки відсутні?
9. Які алгоритми заміщення комірок кеш-пам'яті вам відомі?

6.2 Алгоритми заміщення рядків кеш-пам'яті.

Мета – вивчення впливу параметрів кеш-пам'яті і вибраного алгоритму заміщення на ефективність роботи системи.

Ефективність у даному випадку оцінюється числом кеш-влучень по відношенню до загального числа звернень до пам'яті. Враховуючи різницю в

алгоритмах в режимах наскрізного і зворотнього запису, ефективність використання кеш-пам'яті обчислюється виразами (1.1) і (1.2) відповідно для наскрізного і зворотнього запису (розділ 1.8).

Очевидно, ефективність роботи системи з кеш-пам'яттю буде залежати не тільки від параметрів кеш-пам'яті і вибраного алгоритму заміщення, але і від класу задачі. Так, лінійні програми повинні добре працювати з алгоритмами заміщення типу чергу, а програми з великим числом умовних переходів, що залежать від випадкових вхідних даних, можуть давати непогані результати з алгоритмами випадкового заміщення. Можна припустити, що програми, що мають велике число повторюваних ділянок (часто викликаються підпрограм та/або циклів) при інших рівних умовах забезпечать більш високу ефективність застосування кеш-пам'яті, ніж лінійні програми. І, зрозуміло, на ефективність безпосередньо повинен впливати розмір кеш-пам'яті.

Для перевірки висловлених вище припущень виконується дана лабораторна робота.

6.3 Завдання 2

В даній частині лабораторної роботи всі варіанти однакові завдання: дослідити ефективність роботи кеш-пам'яті при виконанні двох різнотипних програм, написаних і налагоджених вами при виконанні лабораторної роботи № 2 і 3.

6.3.1 Порядок виконання роботи

1. Завантажити модель навчальної ЕОМ налагоджену програму з лабораторної роботи № 2.
2. У меню Робота встановити режим Кеш-пам'ять.
3. У меню Вид вибрати команду Кеш-пам'ять, відкривши тим самим вікно Кеш пам'ять, в ньому натиснути першу зліва кнопку на панелі інструментів, відкривши діалогове вікно Параметри кеш-пам'яті і встановити наступні параметри кеш-пам'яті: розмір - 4, режим запису – наскрізна, алгоритм заміщення - випадкове, без урахування біта запису (W).
4. Запустити програму в автоматичному режимі; по закінченні роботи переглянути результати роботи кеш-пам'яті у вікні Кеш-пам'ять, обчислити значення коефіцієнта ефективності і записати в комірку таблиці 6.3, позначену зірочкою.
5. Вимкнути кеш-пам'ять моделі (Робота | Кеш-пам'ять) і змінити один з її параметрів - встановити прапор з урахуванням біта запису (у вікні Параметри кеш-пам'яті).
6. Повторити п. 4, помістивши значення отриманого коефіцієнта ефективності у наступну комірку праворуч.
7. Послідовно змінюючи параметри кеш-пам'яті, повторити пп. 3-5, заповнюючи всі комірки таблиці 6.3.

8. Повторити всі дії, описані в пп. 1-7 для програми з лабораторної роботи № 4, заповнюючи другу таблицю за формою таблиці 6.3.

6.3.2 Зміст звіту

Дві таблиці за формою таблиці 6.3 з результатами моделювання програм з лабораторних робіт № 2 і 3 при різних режимах роботи кеш-пам'яті.

Висновки, що пояснюють отримані результати

6.3.3 Контрольні питання

1. Як працює алгоритм заміщення черга при встановленому прапорці з урахуванням біта запису в діалоговому вікні Параметри кеш- пам'яті?
2. Який алгоритм заміщення буде найбільш ефективним у разі застосування кеш-пам'яті великого обсягу (в кеш-пам'яті цілком поміщається програма)?
3. Як позначиться на ефективності алгоритмів заміщення облік значення біта запису W при роботі кеш-пам'яті в режимі зворотнього запису? Наскрізного запису?
4. Для яких цілей в структуру комірки кеш-пам'яті включений біт використання. Як встановлюється і скидається цей біт?

Таблиця 6.3 – Результати експерименту

Спосіб	Наскрізний запис					
Алгоритм	Випадковий запис		Черга		Біт U	
Розмір	без W	з W	без W	з W	без W	з W
4						
8						
16						
32						
Спосіб	Зворотній запис					
Алгоритм	Випадковий запис		Черга		Біт U	
Розмір	без W	з W	без W	з W	без W	з W
4						
8						
16						
32						

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Акушский І. Я., Юдицький Д. І. Машинна арифметика в залишкових класах. — М: Сов. радіо, 1968.

2. Баранов С. В. Синтез мікропрограмних автоматів. — Л.: Енергія, 1974.

3. Григор'єв Ст. Л. Мікропроцесор і486. Архітектура та програмування. В чотирьох книгах. — М.: ГРАНАЛ, 1993.

4. Жмакин А. П. Архітектура ЕОМ. — СПб.:БХВ-Петербург, 2006.

5. Жмакин А. П., Тітов В. С. Однокристальні мікроеом в системах управління: Навчальний посібник. — Курськ: Курськ, держ. тех. ун-т, 2002.

6. Закревський А. Д. Алгоритми синтезу дискретних автоматів. — М: Наука, 1971.

7. Каган Б. М. ЕОМ і системи. — М: Вища Школа, 1985.

8. Майоров С. А., Новіков Р. А. Принципи організації ЦВМ.— Л.: Машинобудування, 1974.

9. Савельєв А. Я. Прикладна теорія цифрових автоматів. — М: Вища школа, 1987.

10. Соловйов Р. Н. Арифметичні пристрої ЦОМ. — М: Енергія, 1978.

11. Стариченко Б. Е. Теоретичні основи інформатики.— М.: Гаряча лінія — Телеком, 2003.

12. Таненбаум Е Архітектура комп'ютера. 4-е изд. — СПб.: Пітер, 2003.

13. Хамахер К., Вранешич З., Заки С. Організація ЕОМ. 5-е изд.— СПб.: Пітер, 2003.

14. Хвощ С. Т. та ін Мікропроцесори і мікроеом в системах автоматичного управління: Довідник. — Л.: Машинобудування, 1987.

15. Юров В. І. Assembler. — СПб.: Пітер, 2003.