ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ

Кафедра «Транспортний зв'язок»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання контрольної роботи з дисципліни

"МІКРОПРОЦЕСОРИ В СИСТЕМАХ ТЕЛЕКОМУНІКАЦІЙ"

Харків 2010

Методичні вказівки розглянуто та рекомендовано до друку на засіданні кафедри "Транспортний зв'язок" 24 листопада 2009 р., протокол № 4.

Рекомендуються для студентів факультету "Автоматика, телемеханіка і зв'язок" заочної форми навчання.

Укладач

доц. М.О.Колісник

Рецензент

проф. В.С. Коновалов

МЕТОДИЧНІ ВКАЗІВКИ

до виконання контрольної роботи
з дисципліни
"МІКРОПРОЦЕСОРИ В СИСТЕМАХ ТЕЛЕКОМУНІКАЦІЙ"

Відповідальний за випуск Колісник М.О.

Редактор Буранова Н.В.

Підписано до друку 25.01.10 р.

Формат паперу 60x84 1/16 . Папір писальний.

Умовн.-друк.арк. 2,25. Обл.-вид.арк. 2,5.

Замовлення № Тираж 150. Ціна

Видавництво УкрДАЗТу, свідоцтво ДК № 2874 від. 12.06.2007 р.
Друкарня УкрДАЗТу,
61050, Харків - 50, майдан Фейєрбаха, 7

УКРАЇНСЬКА ДЕРЖАВНА АКАДЕМІЯ
ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ

Кафедра "Транспортний зв'язок"

МЕТОДИЧНІ ВКАЗІВКИ
для виконання контрольної роботи
з дисципліни
"Мікропроцесори в системах телекомунікацій"

для студентів факультету
"Автоматика, телемеханіка і зв'язок" заочної форми навчання

Харків 2009

Методичні вказівки розглянуто та рекомендовано до друку на засіданні кафедри "Транспортний зв'язок" 24 листопада 2009 р., протокол № 4.

Укладач:
доц., к.т.н. Колісник М.О.

Призначено для студентів факультету АТЗ заочної форми навчання.
Табл. 12, бібліогр.: 9

Рецензент:
проф., к.т.н. В.С. Коновалов

ВСТУП

Широке застосування нової техніки з використанням цифрових технологій в усіх галузях науки і виробництва, особливо в галузі телекомунікацій, неможливе без використання мікропроцесорів і вимагає підготовки фахівців якісно нового рівня, які спроможні ефективно використовувати одержані знання та практичні навички програмування і технічного обслуговування такої техніки. Найважливішим пристроєм кожної цифрової системи телекомунікацій є керуючий пристрій, основною складовою частиною якого є мікропроцесор.

Цифрові системи телекомунікацій на основі мікропроцесорів мають невеликі габарити, незначну питому потужність, високу швидкодію, значний рівень надійності, дозволяє реалізовувати алгоритми великої розмірності при невеликих витратах.

Використання мікропроцесорних надвеликих інтегральних схем (НВІС) змінило методи проектування і конструювання засобів обчислювальної техніки, розширило галузі їх використання і коло людей, що займаються розробкою і експлуатацією цих засобів.

Сучасному спеціалісту в галузі телекомунікацій необхідно знати принципи організації і проектування мікропроцесорних обчислювальних і керуючих пристроїв систем телекомунікацій, особливості їх програмування і технічного обслуговування.

Дані методичні вказівки розроблені з метою поглиблення та закріплення знань студентів щодо принципів побудови телекомунікаційних засобів на основі мікропроцесорів, їх програмування, а також набуття практичних навичок складання програм на пересилання даних.

МЕТА РОБОТИ

Закріпити теоретичні знання і набути практичних навичок та уміння в програмуванні мовою нижчого рівня Assembler, у складанні програм на пересилання даних.

ЗАВДАННЯ ДО КОНТРОЛЬНОЇ РОБОТИ

1 За літературою, конспектом лекцій і даними методичними вказівками вивчити принципи, методи і практичні прийоми розроблення програм на пересилання

даних з чарунок оперативної пам’яті та регістрів мікропроцесорної пам’яті.

2 За заданим згідно з індивідуальним варіантом ланцюгом здійснити пересилання даних з чарунок пам’яті та регістрів мікропроцесорної пам’яті.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Процесор – функціонально закінчений, програмно керований пристрій оброблення інформації, виконаний у вигляді однієї або кількох великих (ВІС) та надвеликих (НВІС) інтегральних схем [1].

Мікропроцесор (МП) – це процесор, складові частини якого мініатюризовані та розміщені в одній чи декількох інтегрованих мікросхемах [1].

Функціонально МП можна розділити на дві частини: 1) **операційну частину**, що складається з пристрою управління (ПУ), арифметико-логічного пристрою (АЛП) і мікропроцесорної пам’яті (МПП) (за виключенням деяких адресних регістрів); 2) **інтерфейсну частину**, що містить адресні регістри МПП, блок регістрів команд (регістри пам’яті для зберігання кодів команд, що виконуються в найближчі такти), схеми керування шиною і портами.

Обидві частини МП функціонують паралельно, причому інтерфейсна частина випереджає операційну, так що вибірка наступної команди з пам’яті (її запис у блок регістрів команд і попередній аналіз) виконується у час виконання операційною частиною попередньої команди.

Пристрій управління є функціонально найскладнішим пристроєм мікропроцесорної системи (МПС) – він виробляє керуючі сигнали, що поступають по кодових шинах інструкцій. У склад ПУ входять: регістр команд; дешифратор операцій; постійний запам’ятовуючий пристрій мікропрограм; вузол формування адреси; кодові шини даних, інструкцій та адреси.

Арифметико-логічний пристрій призначений для виконання арифметичних і логічних операцій перетворення інформації. Функціонально АЛП складається з двох регістрів, суматора і схем управління.

Мікропроцесорна пам'ять (МПП) МП служить для тимчасового зберігання даних і результатів виконання логічних та арифметичних операцій. У склад МПП МП, що розглядається в даних методичних вказівках, входять 7 регістрів загального і 3 регістри спеціального призначення.

Інтерфейсна система МП призначена для зв'язку і узгодження МП з системною шиною МПС, а також для приймання, попереднього аналізу команд програми, що виконується, і формування повних адрес операндів і команд. У склад її входять: адресні регістри МПП; вузол формування адреси; блок регістрів команд, що є буфером команд у МП; внутрішня інтерфейсна шина МП; схеми керування шиною і портами вводу-виведення.

Команди мікропроцесора забезпечують виконання операцій над одним або двома операндами, і результат операції може записуватись за адресою будь-якого з операндів. Залежно від типу команди операнди можуть бути розташовані в програмно-доступних регістрах, безпосередньо в коді команди, в пам'яті і регістрах вводу-виведення. Регістри – це швидкодіючі чарунки пам'яті різної довжини, що служать для тимчасового зберігання даних. Безпосередні дані можуть бути типу байта або слова. Операнди в програмно-доступних регістрах можуть бути типу байта або слова, а для команд множення та ділення – типу подвійного слова.

Операнди в пам'яті можуть бути типу байта слова, подвійного слова, а в регістрах вводу-виведення – типу байта і слова.

Адресні регістри або вказівники використовуються для реалізації тих чи інших методів адресації операндів, що використовуються в конкретних командах МП. Їх конкретний

набір і функції залежать від того, які методи адресації реалізовані в даній моделі МП.

Під ***методом адресації*** розуміється метод кодування адреси операнда або результату операції в коді команди.

В загальному випадку код команди МП можна зобразити у вигляді:

КОП	АОП1	АОП2	…	АР

де КОП – код операції; АОП1 – поле адреси першого операнда;

АОП2 – поле адреси другого операнда;

АР – поле адреси результату.

Наявність окремих полів, окрім КОП, визначається конкретною командою і типом МП. Інформація в полях АОП і АР визначається конкретним методом адресації, що використовується в даній команді.

Найбільш поширеними методами адресації, що використовуються в сучасних моделях МП, є:

Регістрова адресація. Операнд знаходиться в регістрі. Адреса регістра включена в код операції. Поле адреси в команді відсутнє.

Пряма адресація. Фізична адреса операнда знаходиться у відповідному полі адреси.

Безпосередня адресація. Безпосереднє значення операнда розташоване у відповідному полі адреси.

Непряма регістрова адресація. Фізична адреса операнда знаходиться в регістрі непрямої адреси **DP** (Data Pointer). Адреса регістра включена в код операції. Поле адреси в команді відсутнє. Як DP може виступати регістр загального призначення (РЗП) або спеціальний адресний регістр.

Залежно від того, які методи адресації реалізовані в конкретному процесорі, в ньому є ті або інші адресні регістри.

Для вказівки операнда в програмно-доступних регістрах використовуються регістрова і неявна регістрова адресація.

Для керування процесом виконання програми використовується слово-стан програми (**PSW** – Program Status Word). Слово стану **PSW** містить набір поточних ознак результату виконання операції. Старший байт слова-стану представляє дані, що містяться в акумуляторі, а молодший – містить прапори умов регістра ознак, що визначаються результатом виконання арифметичних і логічних операцій:

SF	ZF	○	AF	○	PF	○	CF

де: **SF** (Sign Flag) – прапор знака результату. Дублює знаковий розряд результату операцій. Встановлюється в 1, якщо знаковий біт результату операції є від'ємним;

ZF (Zero Flag) – прапор ознаки нуля. Встановлюється в 1, якщо результат операції дорівнює 0;

AF (Auxilinary Carry Flag) – прапор додаткового переносу. Встановлюється в 1, якщо в результаті виконання арифметичної операції або операції зсуву здійснився перенос з молодшої тетради результату в старшу. Часто використовується в двійково-десятковій арифметиці;

PF (Parity Flag) – прапор ознаки парності. Встановлюється в 1, якщо число одиниць у результаті операції непарне і навпаки;

CF (Carry Flag) – прапор переносу зі старшого розряду арифметико-логічного пристрою. Встановлюється в 1, якщо в результаті виконання арифметичної операції або операції зсуву трапився перенос зі старшого розряду результату.

Конкретні прапори використовуються програмою для аналізу результату попередньої команди і прийняття рішення при подальшому ході виконання програми.

Стан прапорів (умов) використовується для реалізації команд умовної передачі керування. Якщо задана в команді умова виконується, то здійснюється передача керування за вказаною адресою. В протилежному випадку – передача керування не здійснюється, а виконується наступна по порядку команда. Основою програмної моделі МП є регістри.

Регістри мікропроцесорної пам'яті МП функціонально неоднорідні: одні служать для зберігання даних або адресної інформації, інші – для керування роботою центрального процесора (ЦП). Відповідно до цього всі регістри можна розділити на регістри даних, вказівники і регістри спеціального призначення. Регістри даних беруть участь в арифметичних і логічних операціях як джерела операндів і приймачів результату, адресні регістри або вказівники використовуються для обчислення адрес даних і команд, що розташовані в основній пам'яті. Спеціальні регістри служать для індикації поточного стану ЦП і керування роботою його складових частин. Можлива архітектура, при якій одні й ті самі регістри використовуються для зберігання як даних, так і адресної інформації. Такі регістри є РЗП. Способи використання того чи іншого виду регістрів визначають конкретні особливості архітектури МП.

Першу групу регістрів утворюють РЗП **A, B, C, D, H, L**. Такими літерами ідентифікуються регістри при програмуванні мовою Assembler. У процесорі існують також програмно недоступні додаткові 8-розрядні регістри акумулятора, тимчасового зберігання даних і два додаткових регістра команд. Серед регістрів даних часто виділяють один регістр, що називається **акумулятором** (**A**) (Accumulator), з яким пов'язують більшість команд арифметичної і логічної обробки даних. Це означає, що

арифметичні і логічні команди використовують як один зі своїх операндів дані, що містяться в акумуляторі, і зберігають у ньому результат операції. Посилання на нього виконується неявно за допомогою коду операції. При цьому немає необхідності в коді команди виділяти спеціальну область для адрес операнда і результату. Такий тип архітектури МП називається **акумуляторним**. До недоліків такої архітектури можна віднести відносно низьку швидкодію, що пояснюється тим, що **A** є "вузьким місцем", в яке кожного разу необхідно спочатку занести операнд перед виконанням операції. Прикладом такої архітектури можуть служити мікроконтролери сімейства MCS-51 фірми Intel. Для МП з акумуляторною архітектурою, що розглядається в даних методичних вказівках, найважливішим та більш використовуваним регістром є **A**. При виконанні операцій за командами МП використовується операнд, що знаходиться в цьому регістрі, а результат розташовується в цьому ж регістрі. З акумулятором найбільш тісно пов'язаний спеціалізований **регістр прапорів** (умов, ознак) **F**.

Архітектура, при якій МП здатний використовувати за адреси операндів і результатів операції чарунки основної пам'яті, називається архітектурою типу "**пам'ять - пам'ять**". При цьому виключаються часові витрати на перепис даних, що містяться в робочих регістрах при переході від однієї процедури до іншої. Однак при цьому губиться швидкий доступ до проміжних даних, бо вони зберігаються не у внутрішніх регістрах, а в просторі даних оперативної пам'яті. Прикладом такої організації можуть служити мікроконтролери сімейства MCS-96 фірми Intel.

Практично в усіх сучасних МП виділяють окрему область пам'яті під так званий "**стек**", що використовується, в загальному випадку, для передачі параметрів процедурам і зберігання адрес повернення з них. Стек може бути розташований всередині МП або зовні. Він може займати частину адресного простору DSEG або RSEG, а може бути розташований і окремо від них. В останньому випадку

кажуть про так званий "***апаратний стек***". Передача функцій від **A** до верхівки стека приводить до "**стекової архітектури**". Стекова організація дає можливість використовувати безадресні команди, код яких має найменшу довжину. Безадресні команди оперують даними, що знаходяться на верхівці стека і безпосередньо під нею. При виконанні операції вихідні операнди витягуються зі стека, а результат передається на вершину стека. Стекова архітектура має високу обчислювальну ефективність.

Службові регістри, розташовані усередині МП, призначені для різних функцій управління його роботою. Їх склад і організація залежать від конкретної архітектури МП і розрізняються в кожному конкретному випадку. Найбільш часто використовуваними регістрами спеціальних функцій є "**програмний лічильник**" **PC** (Program Counter), "**покажчик стека**" **SP** (Stack Pointer) і **"слово стану програми" PSW** (Program Status Word). Програмний лічильник **PC** у кожний конкретний момент часу містить адресу команди, яка переноситься в оперативну пам'ять за тією, яка в даний момент виконується. Покажчик стека **SP** зберігає поточну адресу вершини стека. Слово стану програми **PSW** містить набір поточних ознак результату виконання операції. З кожною ознакою результату зв'язується однорозрядна змінна-прапор, що відповідає визначеному біту PSW. До типових прапорів-ознак належать службові регістри, розташовані всередині МП, призначені для різних функцій керування його роботою та індикації стану його складових частин.

Регістр **F** і регістр **A** для формування слова стану процесора **PSW** утворюють регістрову пару при пересиланні його в стек і зчитуванні зі стека (команди PUSH PSW, POP PSW).

Інші РЗП можуть використовуватись як в однобайтовому варіанті, так і програмно об’єднуватись в регістрові пари **BC, DE, HL**. Пари регістрів можна використовувати для зберігання 16-бітних даних і адресації

даних. Можливий обмін даними між 8-бітними РЗП і **A**, а також між собою. При використанні 16-бітної інформації передбачений обмін лише між регістровими парами **HL** і **DE**, а також між цими парами регістрів, акумулятором і стеком.

Найбільш широко використовується регістрова пара **HL**. Дана пара використовується не тільки за загальним призначенням, а також для вказівки адреси при непрямій адресації (**M** - Memory). В такому випадку пара регістрів **HL** позначається літерою **M** і має трирозрядний номер (адресу) 110. В цій парі регістрів можливо здійснити додавання даних, що містяться в **HL**, до даних, що містяться в іншій парі регістрів. При цьому результат зберігається в **HL** і відповідно до цього встановлюється стан прапора переносу **CF**. Реалізація операції додавання даних, що містяться в парі **HL**, з самими собою подвоює ці дані, що еквівалентно зсуву даних на один розряд вліво.

Іншим прикладом організації регістрів даних є так звані "робочі регістри" RO, R1 і т.д. У цьому випадку операнди і результати арифметичних і логічних операцій можуть зберігатися не в одному, а в кількох регістрах, що розширює можливості щодо маніпуляції даними. На відміну від **A** робочі регістри адресуються явно в коді команди. Такий тип архітектури МП називається **регістровим**. Прикладом такої організації можуть служити МП сімейства 80x86 фірми Intel. У ряді МП, призначених для роботи в реальному масштабі часу, передбачені не один, а кілька наборів робочих регістрів. У кожний момент часу доступний лише один з наборів регістрів, вибір якого здійснюється записом відповідної інформації у визначений службовий регістр. Прикладом таких пристроїв можуть служити мікроконтролери сімейства MCS-48 фірми Intel.

Команди пересилань виконують обмін даними між регістрами загального призначення (РЗП) і пам’яттю МПС. Команди пересилань не впливають на прапори.

Пересилання може здійснюватись:

а) *з пам’яті в акумулятор* (**пряма адресація**).

Команди, які використовуються при прямій адресації:

LDA<ADDR> – запис даних, що містяться в акумуляторі, за адресою, вказаною в другому і третьому байтах команди.

STA<ADDR> – запис даних, що містяться в акумуляторі, в пам'ять за адресою, вказаною в другому і третьому байтах команди.

<**ADDR>** – безпосереднє адресне значення, що може бути константою або міткою в програмі;

б) *з регістра в регістр* (**регістрова адресація**).

Команди, які використовуються при регістровій адресації:

MOV R2, R1 – передача даних, що містяться в регістрі РЗП R2, в регістр РЗП R1.

Команда може бути використовувана для створення копій деяких змінних, які часто використовуються при обчисленнях.

При програмуванні доцільно використовувати саме команди регістрової адресації, бо такі команди займають лише один байт пам’яті. Використання регістрів для зберігання даних виключає обмін з пам’яттю, що зберігає час виконання операцій;

в) з *пам’яті в регістр* (**регістрова непряма адресація**).

Команди, які використовуються при регістровій непрямій адресації:

MOV M, R – передача даних, що містяться в регістрі РЗП R, в пам'ять за адресою, що зберігається в регістровій парі (**H**, **L**).

MOV R, M – передача даних, що містяться в чарунці пам’яті, адреса якої зберігається в регістровій парі (**H**, **L**), в регістр РЗП R.

Ці команди широко використовуються при обробці масивів даних;

г) б*езпосередньо в регістр РЗП байта даних* (**безпосередня адресація**).

Команди, які використовуються при безпосередній адресації:

MVI R <DATA> – передача байта даних у регістр R.

MVI M <DATA> – передача байта даних у чарунку пам’яті, адреса якої зберігається в регістровій парі (**H**, **L**).

LXI R <DATA> – передача даних, що містяться в чарунці пам’яті, адреса якої зберігається в регістровій парі (**H**, **B**, **D**).

<DATA> – безпосереднє значення даних (константа).

Команди безпосередньої адресації містять операнд. Перевагою таких команд є швидкодія та економія об’єму пам’яті мікропроцесорної системи.

Під операндом розуміють число, над яким виконуються арифметичні та логічні операції.

Команди пересилання асемблеру мікропроцесора (МП) І8080, які використовуються при розв’язанні контрольної роботи, наведені в таблиці 1. Повний перелік команд асемблеру МП І8080 можна знайти у [2].

Таблиця 1 – Команди пересилання асемблеру МП І8080

Мнемокод	Довжина байт	Двійковий код	Опис операції	Шістнадцятирічний код
1	2	3	4	5
STA <addr>	3	00110010	Запис інформації, що	

			знаходиться в регістрі МПП акумулятора (A), в чарунку оперативної пам’яті за адресою, що представлена 2 і 3 байтами команди	32
LDA<addr>	3	00111010	Копіювання інформації, що знаходиться в чарунці оперативної пам’яті за адресою, що представлена 2 і 3 байтами команди, в регістр МПП акумулятор (A)	3A
LXI B <addr>	3	00000001	Копіювання інформації, що знаходиться в чарунці оперативної пам’яті за адресою, що представлена 2 і 3 байтами команди, в B-пару регістрів МПП (B,C)	01
LXI D <addr>	3	00010001	Копіювання інформації, що знаходиться в чарунці оперативної пам’яті за адресою, що представлена 2 і 3 байтами команди, в D-пару регістрів МПП (D,E)	11
LXI H <addr>	3	00100001	Копіювання інформації, що знаходиться в чарунці оперативної пам’яті за адресою, що представлена 2 і 3 байтами команди, в H-пару регістрів МПП (H,L)	21

Продовження таблиці 1

1	2	3	4	5
MOV R2,R1	1	01XXXYYY	Пересилання даних з регістра МПП R1 в регістр МПП R2	..
MOV R,M	1	01XXX110	Пересилання даних з чарунки МПП пам’яті, представленої парою регістрів (H,L) в регістр МПП R (A,B,C,D,E)	..
MOV M,R	1	01110YYY	Пересилання даних з	

			регістра МПП R (A,B,C,D,E) в чарунку МПП пам'яті, представленої парою регістрів (H,L)	7
MVI R <data>	2	00XXX110	Безпосереднє пересилання даних (одного байта інформації, представленої шістнадцятирічним кодом) в регістр МПП R	..
MVI M <data>	2	00110110	Безпосереднє пересилання даних (одного байта інформації, представленої шістнадцятирічним кодом) у чарунку МПП пам'яті, представленої парою регістрів (H,L)	36
HLT	1	01110110	Зупинка	76

Двійкові коди кожного з РЗП мікропроцесорної пам'яті МП, що розглядається:

000 – **B**;
001 – **C**;
010 – **D**;
011 – **E;**
100 – **H**;
101 – **L**;
111 – **A**.

В таблиці 1 не наведено шістнадцятирічного коду для команди **MOV**, бо машинні коди при використанні різних регістрів будуть відрізнятись. Наприклад, необхідно знайти шістнадцятирічний код команди **MOV C,A**. В таблиці 2 наведено правило переводу коду з десяткової системи числення у двійкову та шістнадцятирічну для 16 чисел. Кількість розрядів для двійкового зображення визначається для 16 чисел за правилом: $2^n=16$, тоді n = 4.

Таблиця 2 – Правило переводу чисел з однієї системи числення в іншу

Десяткова	Двійкова	Шістнадцятирічна
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

У двійковому коді команда (таблиця1, стовпчик 3) має вигляд:

$$\mathbf{01XXXУУУ_2},$$

де ХХХ відповідає двійковому коду регістра, в який заноситься інформація, УУУ – регістру, з якого пересилається інформація. Команда **MOV C,A** здійснює пересилання даних з регістра **A** в регістр **C**. Для команди **MOV C,A** замість ХХХ підставляємо код **001** (який відповідає регістру **C**), замість УУУ – код **111** (що відповідає регістру **A**). Тоді отримуємо двійковий код команди:

$$\mathbf{01001111_2}.$$

Далі переводимо код з двійкової системи числення в шістнадцятирічну, поділивши двійковий код на тетради (по 4 розряди) і присвоївши кожній тетраді відповідний шістнадцятирічний код :

$$\underbrace{0100}_{4}\underbrace{1111}_{F}{}_2 = \mathbf{4F}_{16.}$$

Тоді для команди **MOV C,A** шістнадцятирічний код – **$4F_{16}$.**

Аналогічно для команди **MOV A,C** шістнадцятирічний код визначається як:

$$\underbrace{01111}_{7}\underbrace{001}_{9}{}_{2} = \mathbf{79}_{16}.$$

Виконаємо як приклад **розв'язання задачі №1**: Здійснити пересилання даних з однієї чарунки пам'яті в іншу і в зворотньому напрямку. Виконати розв'язання для двох ситуацій: 1) одна з чарунок містить дані, інша – порожня; 2) обидві чарунки пам'яті містять дані, тому перед пересиланням інформації дані з них необхідно зберегти в інших чарунках пам'яті або регістрах.

(0901) ⇔(080B).

Розв'язання може мати такий вигляд:

1 Якщо чарунки не містять важливої інформації, яку необхідно зберегти, то пересилання інформації з чарунки (0901) в (080B) і навпаки буде здійснюватись таким чином. Заносимо (копіюємо) дані, що містить чарунка (0901), в акумулятор за допомогою команди LDA. Потім виносимо згідно з вихідними даними інформацію з акумулятора в чарунку за адресою (080B). Після цього повторюємо дану операцію у зворотньому напрямку. Програма пересилання даних показана в таблиці 3.

При цьому ланцюг, який використовується при пересиланні, має такий вигляд: (0901)↓(A)↓(080B)↓(A)↓(0901).

Таблиця 3 – Програма пересилання (0901) ⇔(080B), коли одна з чарунок містить дані, інша – порожня

Адреса	Команда	Шістнадцятирічний код команди	Коментар
0800	LDA 0901	3A	Дані, що містяться в чарунці (0901), пересилаємо в
0801		01	
0802		09	

			акумулятор: (0901)↓(A)
0803	STA 080B	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (080B): (A)↓(080B)
0804		0B	
0805		08	
0806	LDA 080B	3A	Дані, що містяться в чарунці (080B), пересилаємо в акумулятор: (080B)↓(A)
0807		0B	
0808		08	
0809	STA 0901	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0901): (A)↓(0901)
080A		01	
080B		09	
080C	HLT	76	Зупинка

2 Якщо чарунка (080B) містить важливу інформацію, яку необхідно зберегти, то пересилання інформації з чарунки (0901) в (080B) і навпаки буде здійснюватись таким чином. Програму починаємо з адреси, не меншої ніж (0800), бо для МП, що розглядається, адресний простір для запису програм і даних в оперативний запам’ятовуючий пристрій починається з адреси (0800). Заносимо (копіюємо) дані, що містить чарунка (080B), в мікропроцесорну пам'ять (пару регістрів **H**,**L**) за допомогою команди LXI H. Потім повторюємо програму для варіанта 1 (таблиця 4).

При цьому ланцюг операцій, які використовуються при пересиланні, має такий вигляд:

(080B)↓(M(H,L))↓(0901)↓(A)↓(080B)↓(A)↓(0901).

Таблиця 4 – Програма пересилання (0901) ⇔(080B), коли обидві чарунки пам’яті містять дані, тому перед пересиланням інформації дані з них необхідно зберегти в інших чарунках пам’яті або регістрах

Адреса	Команда	Шістнадцяти-річний код команди	Коментар
0902	LXI H 080B	21	Дані, що містяться в чарунці

0903		0B	(080B), пересилаємо в мікропроцесорну пам'ять (M), що представлена регістровою парою (H,L): (080B)↓(M(H,L))
0904		09	
0905	LDA 0901	3A	Дані, що містяться в чарунці (0901), пересилаємо в акумулятор: (0901)↓(A)
0906		01	
0907		09	
0908	STA 080B	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (080B): (A)↓(080B)
0909		0B	
090A		08	
090B	LDA 080B	3A	Дані, що містяться в чарунці (080B), пересилаємо в акумулятор: (080B)↓(A)
090C		0B	
090D		08	
090E	STA 0901	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0901): (A)↓(0901)
090F		01	
0910		09	
0911	HLT	76	Зупинка

При виконанні задачі може бути і таке розв’язання варіанта 2. Інформація з чарунки з адресою (0901) передається в чарунку за адресою (080B), а інформація, що містилась у чарунці з адресою (080B) до пересилання, – в чарунку з адресою (0901). Тоді програма пересилання має такий вигляд (таблиця 5).

Таблиця 5 – Програма пересилання (0901) ⇔(080B), коли обидві чарунки пам’яті містять дані, тому перед пересиланням інформації дані з них необхідно зберегти в інших чарунках пам’яті або регістрах

Адреса	Команда	Шістнадцяти річний код команди	Коментар
0902	LXI H 080B	21	Дані, що містяться в чарунці (080B), пересилаємо в мікропроцесорну пам'ять (M), що представлена регістровою парою (H,L): (080B)↓(M(H,L))
0903		0B	
0904		09	
0905	LDA 0901	3A	Дані, що містяться в чарунці (0901), пересилаємо в акумулятор: (0901)↓(A)
0906		01	
0907		09	
0908	MOV C,A	4F	Пересилання даних, що

			містяться в акумуляторі, в регістр С
0909	MOV A,C	79	Дані, що містяться в регістрі С, копіюємо в акумулятор
090A	STA 080B	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (080B): (A)↓(080B)
090B		0B	
090C		08	
090D	MOV A,M	7E	Дані, що містяться в мікропроцесорній пам'яті (M), яка представлена регістровою парою (H,L), копіюються в акумулятор: (M(H,L))↓A
090E	STA 0901	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0901): (A)↓(0901)
090F		01	
0910		09	
0911	HLT	76	Зупинка

Виконаємо для прикладу **розв'язання задачі №2**: Поміняти місцями дані, що знаходяться в чарунках пам'яті та регістрах, по ланцюгу, поданому для кожного з варіантів. Виконати розв'язання для двох ситуацій: 1) чарунки пам'яті та регістри, крім перших у ланцюгу, містять дані, інші – порожні; 2) чарунки пам'яті та регістри містять дані, тому перед пересиланням інформації дані з них необхідно зберегти в інших чарунках пам'яті або регістрах.

(0901) ↓(0903)↓(080A)↓(0900).

Розв'язання може мати такий вигляд:

1 Якщо чарунки не містять важливої інформації, яку необхідно зберегти, то пересилання інформації з чарунки (0901) в інші буде здійснюватись таким чином. Заносимо (копіюємо) дані, що містить чарунка (0901), в акумулятор за допомогою команди LDA. Потім виносимо згідно з вихідними даними інформацію з акумулятора в чарунку за адресою (0903). Потім повторюємо дану операцію для кожної чарунки у прикладі. Програма пересилання даних подана в таблиці 6.

При цьому ланцюг, який використовується при пересиланні, має такий вигляд:

(0901)↓(A)↓(0903)↓(A)↓(080A)↓(A)↓(0900).

Таблиця 6 – Програма пересилання даних по ланцюгу (0901)↓(0903)↓(080A)↓(0900), коли одна з чарунок (0901) містить дані, інші – порожні

Адреса	Команда	Шістнадцятирічний код команди	Коментар
1	2	3	4
0904	LDA 0901	3A	Дані, що містяться в чарунці (0901), пересилаємо в акумулятор: (0901)↓(A)
0905		01	
0906		09	
0907	STA 0903	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0903): (A)↓(0903)
0908		03	
0909		09	

Продовження таблиці 6

1	2	3	4
090A	LDA 0903	3A	Дані, що містяться в чарунці (0903), копіюємо в акумулятор: (0903)↓(A)
090B		03	
090C		09	
090D	STA 080A	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (080A): (A)↓(080A)
090E		0A	
090F		08	
0910	LDA 080A	3A	Дані, що містяться в чарунці (080A), копіюємо в акумулятор: (080A)↓(A)
0911		0A	
0912		08	
0913	STA 0900	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0900): (A)↓(0900)
0914		00	
0915		09	
0916	HLT	76	Зупинка

2 Якщо чарунки містять важливу інформацію, яку необхідно зберегти, то пересилання інформації з чарунки в

чарунку буде здійснюватись таким чином. Програму починаємо з адреси, не меншої ніж (0800), але такої, щоб адреси програми не перетинались з адресами у вихідних даних, бо для МП, що розглядається, адресний простір для запису програм і даних в оперативний запам’ятовуючий пристрій починається з адреси (0800). Заносимо (копіюємо) дані, що містить чарунка (080B), в мікропроцесорну пам'ять (пару регістрів **H**,**L**) за допомогою команди LXI **H**. Потім повторюємо програму для варіанта 1 (таблиця 7).

При цьому ланцюг операцій, які використовуються при пересиланні, має такий вигляд:

(0903)↓(M(H,L))↓(080A)↓(A)↓(0904)↓(0900)↓(A)↓
(0905)↓ (0901)↓(A)↓(0903)↓(A)↓(080A)↓(A)↓(0900).

Таблиця 7 – Програма пересилання (0901)↓(0903)↓(080A)↓(0900), коли всі чарунки пам’яті містять дані, тому перед пересиланням інформації дані з них необхідно зберегти в інших чарунках пам’яті або регістрах

Адреса	Команда	Шістнадцятирічний код команди	Коментар
1	2	3	4
0906	LXI H 0903	21	Дані, що містяться в чарунці (0903), пересилаємо в мікропроцесорну пам'ять (M), що представлена регістровою парою (H,L): (0903)↓(M(H,L))
0907		03	
0908		09	
0909	LDA 080A	3A	Дані, що містяться в чарунці (080A), пересилаємо в акумулятор: (080A)↓(A)
090A		0A	
090B		08	
090C	STA 0904	32	Дані, що містяться в акумуляторі, пересилаємо для зберігання в чарунку (0904): (A)↓(0904)
090D		04	
090E		09	
090F	LDA 0900	3A	Дані, що містяться в

0910		00	чарунці (0900), пересилаємо в акумулятор: (0900)↓(A)
0911		09	
0912	STA 0905	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0905) для зберігання: (A)↓(0905)
0913		05	
0914		09	
0915	LDA 0901	3A	Дані, що містяться в чарунці (0901), пересилаємо в акумулятор: (0901)↓(A)
0916		01	
0917		09	
0918	STA 0903	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0903): (A)↓(0903)
0919		03	
091A		09	

Продовження таблиці 7

1	2	3	4
091B	LDA 0903	3A	Дані, що містяться в чарунці (0903), копіюємо в акумулятор: (0903)↓(A)
091C		03	
091D		09	
091E	STA 080A	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (080A): (A)↓(080A)
091F		0A	
0920		08	
0921	LDA 080A	3A	Дані, що містяться в чарунці (080A), копіюємо в акумулятор: (080A)↓(A)
0922		0A	
0923		08	
0924	STA 0900	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0900): (A)↓(0900)
0925		00	
0926		09	
0927	HLT	76	Зупинка

При виконанні задачі може бути і таке розв’язання варіанта 2. Інформація з чарунки з адресою (0901) передається в чарунку за адресою (0903), інформація, що містилась у чарунці з адресою (0903) до пересилання, – в чарунку з адресою (080A), інформація, що містилась у чарунці з адресою (080A) до пересилання, – в чарунку з

адресою (0900). Тоді програма пересилання має такий вигляд (таблиця 8).

Таблиця 8 – Програма пересилання (0901)↓(0903)↓(080A)↓(0900), коли всі чарунки пам’яті містять дані, тому перед пересиланням інформації дані з них необхідно зберегти в інших чарунках пам’яті або регістрах

Адреса	Команда	Шістнадцятирічний код команди	Коментар
1	2	3	4
0906	LXI H 0903	21	Дані, що містяться в чарунці (0903), пересилаємо в мікропроцесорну пам'ять (М), що представлена регістровою парою (H,L): (0903)↓(M(H,L))
0907		03	
0908		09	

Продовження таблиці 8

1	2	3	4
0909	LDA 080A	3A	Дані, що містяться в чарунці (080A), пересилаємо в акумулятор: (080A)↓(A)
090A		0A	
090B		08	
090C	STA 0904	32	Дані, що містяться в акумуляторі, пересилаємо для зберігання в чарунку (0904): (A)↓(0904)
090D		04	
090E		09	
090F	LDA 0900	3A	Дані, що містяться в чарунці (0900), пересилаємо в акумулятор: (0900)↓(A)
0910		00	
0911		09	
0912	STA 0905	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0905) для зберігання: (A)↓(0905)
0913		05	
0914		09	
0915	LDA 0901	3A	Дані, що містяться в чарунці (0901), пересилаємо в акумулятор: (0901)↓(A)
0916		01	
0917		09	
0918	STA 0903	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0903):
0919		03	
091A		09	

			(A)↓(0903)
091B	MOV A,M	7E	Дані, що містяться в мікропроцесорній пам'яті (M), що представлена регістровою парою (H,L), копіюємо в акумулятор: (M(H,L))↓(A)
091E	STA 080A	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (080A): (A)↓(080A)
091F		0A	
0920		08	
0921	LDA 0904	3A	Дані, що містяться в чарунці (0904), копіюємо в акумулятор (0904)↓(A).
0922		04	
0923		09	
0924	STA 0900	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0900): (A)↓(0900)
0925		00	
0926		09	
0927	HLT	76	Зупинка

Виконаємо для прикладу **розв'язання задачі №3**: Поміняти місцями дані, що знаходяться в чарунках пам'яті та регістрах, по ланцюгу, поданому для кожного з варіантів. Виконати розв'язання для двох ситуацій: 1) чарунки пам'яті та регістри, крім перших у ланцюгу, містять дані, інші – порожні; 2) всі чарунки пам'яті та регістри містять дані, тому перед пересиланням інформації дані з них необхідно зберегти в інших чарунках пам'яті або регістрах.

(0902)↓(A)↓(0901)↓(0905)↓(B).
↑_____→

Розв'язання може мати такий вигляд.

1) Якщо чарунки не містять важливої інформації, яку необхідно зберегти, то пересилання інформації з чарунки (0902) в інші чарунки та регістри буде здійснюватись таким чином. Заносимо (копіюємо) дані, що містить чарунка (0902), в акумулятор за допомогою команди **LDA**. Потім виносимо згідно з вихідними даними інформацію з акумулятора в чарунку за адресою (0901). Потім

повторюємо дану операцію для кожної чарунки у прикладі. Програма пересилання даних подана в таблиці 9.

При цьому ланцюг, який використовується при пересиланні, має такий вигляд:

(0902)↓(A)↓(0901)↓(A)↓(0905)↓(A)↓(B)↓(0901).

Таблиця 9 – Програма пересилання даних по ланцюгу (0902)↓(A)↓(0901)↓(0905)↓(B), коли одна з

↑_____→

чарунок (0902) містить дані, інші – порожні

Адреса	Команда	Шістнадцятирічний код команди	Коментар
1	2	3	4
0906	LDA 0902	3A	Дані, що містяться в чарунці (0902), пересилаємо в акумулятор: (0902)↓(A)
0907		02	
0908		09	

Продовження таблиці 9

1	2	3	4
0909	STA 0901	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0901): (A)↓(1)
090A		01	
090B		09	
090C	LDA 0901	3A	Дані, що містяться в чарунці (0901), копіюємо в акумулятор: (0901)↓(A)
090D		01	
090E		09	
090F	STA 0905	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0905): (A)↓(0905)
0910		05	
0911		09	
0912	LDA 0905	3A	Дані, що містяться в чарунці (0905), копіюємо в акумулятор: (0905)↓(A)
0913		05	
0914		09	
0915	MOV B,A	47	Перенесення даних з регістра (A) в регістр (B)
0916	MOV A,B	78	Перенесення даних з регістра (B) в регістр (A)

0917	STA 0901	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0901): (A)↓(0901)
0918		01	
0919		09	
091A	HLT	76	Зупинка

2) Якщо чарунки (080B) містять важливу інформацію, яку необхідно зберегти, то пересилання інформації з чарунки (0901) в (080B) і навпаки буде здійснюватись таким чином. Програму починаємо з адреси не меншої, ніж (0800), бо для МП, що розглядається, адресний простір для запису програм і даних в оперативний запам'ятовуючий пристрій починається з адреси (0800). Заносимо (копіюємо) дані, що містить чарунка (080B), в мікропроцесорну пам'ять (пару регістрів **H**,**L**) за допомогою команди **LXI H<ADDR>**. Потім повторюємо програму для варіанта 1 (таблиця 10).

При цьому ланцюг етапів операцій, які використовуються при пересиланні, має такий вигляд:

(A)↓(C)↓(B)↓(D)↓(0901)↓(M(H,L))↓(0905)↓(A)↓(0907)↓
(0902)↓(A)↓(0901)↓
(0901)↓(A)↓(0905)↓(A)↓(B)↓(A)↓(0901).

Таблиця 10 – Програма пересилання даних по ланцюгу (0902)↓(A)↓(0901)↓(0905)↓(B), коли всі ↑——→ чарунки пам'яті містять дані, тому перед пересиланням інформації дані з них необхідно зберегти в інших чарунках пам'яті або регістрах

Адреса	Команда	Шістнадцятирічний код команди	Коментар
1	2	3	4
0800	MOV C,A	4F	Перенесення даних з регістра (A) в регістр (C)
0801	MOV D,B	57	Перенесення даних з регістра (B) в регістр (D)
0802	LXI H 0901	21	Дані, що містяться в чарунці (0901), пересилаємо в
0803		01	
0804		09	

			мікропроцесорну пам'ять (M), що представлена регістровою парою (H,L): (0903)↓(M(H,L))
0805	LDA 0905	3A	Дані, що містяться в чарунці (080A), пересилаємо в акумулятор: (080A)↓(A)
0806		05	
0807		09	
0808	STA 0907	32	Дані, що містяться в акумуляторі, пересилаємо для зберігання в чарунку (0907): (A)↓(0907)
0809		07	
080A		09	
080B	LDA 0902	3A	Дані, що містяться в чарунці (0902), пересилаємо в акумулятор: (0902)↓(A)
080C		02	
080D		09	

Продовження таблиці 10

1	2	3	4
080E	STA 0901	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0901) для зберігання: (A)↓(0901)
080F		01	
0810		09	
0811	LDA 0901	3A	Дані, що містяться в чарунці (0901), пересилаємо в акумулятор: (0901)↓(A)
0812		01	
0813		09	
0814	STA 0905	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0905): (A)↓(0905)
0815		05	
0816		09	
0817	LDA 0905	3A	Дані, що містяться в чарунці (0905), копіюємо в акумулятор: (0905)↓(A)
0818		05	
0819		09	
081A	MOV B,A	47	Перенесення даних з регістра (A) в регістр (B)
081B	MOV A,B	78	Перенесення даних з регістра (B) в регістр (AB)
081C	STA 0900	32	Дані, що містяться в акумуляторі, пересилаємо
081D		00	

081E		09	в чарунку (0900): (A)↓(0900)
081F	HLT	76	Зупинка

При виконанні задачі може бути і таке розв'язання варіанта 2. Інформація з чарунки з адресою (0902) передається в регістр **(A)**, інформація, що містилась в регістрі **(A)** до пересилання, – в чарунку з адресою (0905), інформація, що містилась в чарунці з адресою (0905) до пересилання, – в регістр **(B)**. Тоді програма пересилання має такий вигляд (таблиця 11).

Таблиця 11 – Програма пересилання даних по ланцюгу (0902)↓(A)↓(0901)↓(0905)↓(B), коли всі ↑_____→ чарунки пам'яті містять дані, тому перед пересиланням інформації дані з них необхідно зберегти в інших чарунках пам'яті або регістрах

Адреса	Команда	Шістнадцятирічний код команди	Коментар
0800	MOV C,A	4F	Перенесення даних з регістра (A) в регістр (C)
0801	MOV D,B	57	Перенесення даних з регістра (B) в регістр (D)
0802	LXI H 0901	21	Дані, що містяться в чарунці (0901), пересилаємо в мікропроцесорну пам'ять (M), що представлена регістровою парою (H,L): (0901)↓(M(H,L))
0803		01	
0804		09	
0805	LDA 0905	3A	Дані, що містяться в чарунці (0905), пересилаємо в акумулятор: (0905)↓(A)
0806		05	
0807		09	
0808	STA 0907	32	Дані, що містяться в

0809		07	акумуляторі, пересилаємо для зберігання в чарунку (0907): (A)↓(0907)
080A		09	
080B	LDA 0902	21	Дані, що містяться в чарунці (0902), пересилаємо в акумулятор: (0902)↓(A)
080C		02	
080D		09	
080E	MOV A,C	79	Перенесення даних з регістра (A) в регістр (C)
0810	STA 0901	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0901): (A)↓(0901)
0811		01	
0812		09	

Продовження таблиці 11

1	2	3	4
0813	MOV A,M	7E	Дані, що містяться в мікропроцесорній пам'яті (M), що представлена регістровою парою (H,L), копіюємо в акумулятор: (M(H,L))↓(A)
0814	STA 0905	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0905): (A)↓(0905)
0815		05	
0816		09	
0817	LDA 0907	3A	Дані, що містяться в чарунці (0907), копіюємо в акумулятор: (0907)↓(A)
0818		07	
0819		09	
081A	MOV B,A	47	Перенесення даних з регістра (A) в регістр (B)
081B	MOV A,D	7A	Перенесення даних з регістра (D) в регістр (A)
081C	STA 0901	32	Дані, що містяться в акумуляторі, пересилаємо в чарунку (0901): (A)↓(0901)
0925		01	
0926		09	
0927	HLT	76	Зупинка

КОНТРОЛЬНІ ПИТАННЯ

1 Дайте визначення мікропроцесора.
2 Назвіть функціональні складові частини мікропроцесора.
3 Поясніть, для чого необхідні команди мікропроцесора і які команди існують.
4 Назвіть основні методи адресації мікропроцесора.
5 Які існують архітектури мікропроцесорів? Назвіть особливості кожної з них.
6 Для чого використовується АЛП?
7 Назвіть складові частини пристрою управління МП.
8 Що мається на увазі під інтерфейсною системою МП?
9 Для чого необхідні команди пересилання?
10 Назвіть команди пересилання, що використовуються при кожному виді адресації даних МП.
11 Виконайте розв'язання задачі, яку видає викладач з повним описом програми пересилання даних.
12 Назвіть найважливіший регістр МПП і його функції.
13 Для чого необхідне слово стану програми?
14 Для чого формуються прапори умов при виконанні деяких арифметичних і логічних операцій?
15 Для чого використовуються регістри МПП?
16 Що таке стек і які його функції?

ІНДИВІДУАЛЬНІ ЗАВДАННЯ

Варіант індивідуального завдання до контрольної роботи обирається за списком у журналі викладача. Варіанти наведені в таблиці 12.

Задача 1

Здійснити пересилання даних з однієї чарунки пам'яті в іншу і навпаки. Виконати розв'язання для двох ситуацій: 1) одна з чарунок містить дані, інша – порожня; 2) обидві чарунки пам'яті містять дані, тому перед пересиланням

інформації дані з них необхідно зберегти в інших чарунках пам’яті або регістрах.

Задача 2

Поміняти місцями дані, що знаходяться в чарунках пам’яті та регістрах, по ланцюгу, поданому для кожного з варіантів. Виконати розв’язання для двох ситуацій: 1) чарунки пам’яті та регістри, крім перших у ланцюгу, містять дані, інші – порожні; 2) чарунки пам’яті та регістри містять дані, тому перед пересиланням інформації дані з них необхідно зберегти в інших чарунках пам’яті або регістрах.

Задача 3

Поміняти місцями дані, що знаходяться в чарунках пам’яті та регістрах, по ланцюгу, поданому для кожного з варіантів. Виконати розв’язання для двох ситуацій: 1) чарунки пам’яті та регістри, крім перших у ланцюгу, містять дані, інші – порожні; 2) всі чарунки пам’яті та регістри містять дані, тому перед пересиланням інформації дані з них необхідно зберегти в інших чарунках пам’яті або регістрах.

Таблиця 12

В – 1	**В - 2**
1) (0810) ⇔(0900); 2) (080A)↓(A)↓(090B)↓(B); 3) (B)↓(0901)↓(C)↓(A). ↑_____→	1) (090A) ⇔(0801); 2) (0900)↓(B)↓(0901)↓(C); 3) (D)↓(C)↓(A)↓(0902). ↑_____→
В - 3	**В – 4**
1) (0809) ⇔(0813); 2) (0800)↓(0901)↓(B)↓(D);	1) (0805) ⇔(0902); 2) (0801)↓(0802)↓(0803)↓(C);

3) (0801)↓(B)↓(C)↓(0802). ↑_____→	3) (0900)↓(0902)↓(D) ↓(A). ↑_____→
B - 5	**B - 6**
1) (0900) ⇔(0901); 2) (0800)↓(090B)↓(B)↓(090A); 3) (B)↓(A)↓(0907)↓(H). ↑_____→	1) (080E) ⇔(090A); 2) (0800)↓(C)↓(D)↓(A); 3) (C)↓(B)↓(0800)↓(D). ↑_____→

Продовження таблиці 12

B - 7	**B - 8**
1) (0815) ⇔(0910); 2) (0801)↓(0802)↓(E)↓(0803); 3) (0802)↓(0801)↓(A)↓(0903). ↑_____→	1) (0915) ⇔(0807); 2) (D)↓(0801)↓(A)↓(0803); 3) (D)↓(E)↓(0802)↓(C). ↑_____→
B - 9	**B - 10**
1) (0808) ⇔(0901); 2) (0805)↓(D)↓(0800↓(0900); 3) (0802)↓(0801)↓(D)↓(C). ↑_____→	1) (0801) ⇔(0900); 2) (0802)↓(A)↓(0801)↓(0800); 3) (0800)↓(D)↓(B)↓(H). ↑_____→
B - 11	**B - 12**
1) (0800) ⇔(0901); 2) (D)↓(E)↓(080A)↓(A); 3) (E)↓(B)↓(H)↓(A). ↑_____→	1) (080E) ⇔(0900); 2) (0901)↓(C)↓(B)↓(E); 3) (0800)↓(H)↓(0801)↓(D). ↑_____→
B - 13	**B - 14**
1) (0902) ⇔(080C);	1) (0906) ⇔(0817);

2) (H)↓(0802)↓(A)↓(0901); 3) (0902)↓(0903)↓(E)↓(C). ↑_____→	2) (A)↓(0801)↓(0802)↓(D); 3) (0801)↓(0802)↓(C)↓(D). ↑_____→

Продовження таблиці 12

B - 15	**B - 16**
1) (0803) ⇔(090F); 2) (0800)↓(0900)↓(0901)↓(H); 3) (B)↓(A)↓(0902)↓(D). ↑_____→	1) (0901) ⇔(0902); 2) (0801)↓(0903)↓(B)↓(D); 3) (0901)↓(B)↓(0902)↓(E). ↑_____→
B - 17	**B - 18**
1) (0903) ⇔(0905); 2) (0901)↓(B)↓(H)↓(0903); 3) (E)↓(0802)↓(0901)↓(D). ↑_____→	1) (080C) ⇔(080A); 2) (D)↓(0901)↓(H)↓(0902); 3) (B)↓(A)↓(0904)↓(0905). ↑_____→
B - 19	**B - 20**
1) (0801) ⇔(0820); 2) (B)↓(0903)↓(0817)↓(H); 3) (D)↓(0915)↓(0800)↓(C). ↑_____→	1) (083A) ⇔(0900); 2) (0807)↓(0817)↓(081A)↓(D); 3) (C)↓(E)↓(0800)↓(H)↓(0901). ↑_____→

В - 21	**В - 22**
1) (0809) ⇔(092B); 2) (0901)↓(0803)↓(H)↓(C); 3) (A)↓(0901)↓(0902)↓(D). ↑_____→	1) (0807) ⇔(0913); 2) (0805)↓(0909)↓(B)↓(E); 3) (D)↓(0901)↓(A)↓(0903)↓(H). ↑_____→

Продовження таблиці 12

В - 23	**В - 24**
1) (0901) ⇔(0903); 2) (083B)↓(E)↓(080C)↓(D); 3) (091A)↓(0804)↓(D)↓(H). ↑_____→	1) (0907) ⇔(080A); 2) (0801)↓(0807)↓(H)↓(A); 3) (080B)↓(0900)↓(D)↓(C). ↑_____→
В - 25	**В - 26**
1) (0800) ⇔(091B); 2) (0903)↓(B)↓(H)↓(0907)↓(A); 3) (0801)↓(0905)↓(C)↓(D). ↑_____→	1) (0809) ⇔(0813); 2) (0800)↓(0901)↓(B)↓(D); 3) (0801)↓(B)↓(C)↓(0802). ↑_____→
В - 27	**В - 28**
1) (0820) ⇔(0902); 2) (0902)↓(B)↓(0903)↓(D); 3) (0800)↓(0802)↓(0803)↓(D). ↑_____→	1) (0900) ⇔(0801); 2) (0901)↓(C)↓(H)↓(0801); 3) (0802)↓(A)↓(0901)↓(0803). ↑_____→
В - 29	**В - 30**

1) (0819) ⇔(0815); 2) (0905)↓(E)↓(B)↓(A); 3) (B)↓(D)↓(0903)↓(0904). ↑_____→	1) (0903) ⇔(0800); 2) (0900)↓(0815)↓(0800)↓(B); 3) (0901)↓(D)↓(A)↓(0902). ↑_____→

Продовження таблиці 12

В - 31	**В - 32**
1) (0913) ⇔(0915); 2) (0801)↓(D)↓(0802)↓(0803); 3) (0815)↓(A)↓(0915)↓(B). ↑_____→	1) (0803) ⇔(0805); 2) (0901)↓(0815)↓(0816)↓(D); 3) (0812)↓(D)↓(H)↓(0900). ↑_____→
В - 33	**В - 34**
1) (0903) ⇔(0904); 2) (0914)↓(0916)↓(B)↓(C); 3) (0900)↓(090A)↓(D)↓(A). ↑_____→	1) (080A) ⇔(0901); 2) (090C)↓(080B)↓(0801)↓(E); 3) (A)↓(0902)↓(B)↓(0903). ↑_____→
В - 35	**В - 36**
1) (0902) ⇔(0802); 2) (0801)↓(0802)↓(0800)↓(C); 3) (090C)↓(E)↓(A)↓(0900). ↑_____→	1) (0801) ⇔(080B); 2) (0903)↓(0904)↓(0801)↓(E); 3) (0800)↓(0900)↓(E)↓(D)↓(A).
В - 37	**В - 38**
1) (0801) ⇔(090A); 2) (0807)↓(0914)↓(080A)↓(E);	1) (0805) ⇔(080E); 2) (090F)↓(0807)↓(D)↓(C);

3) (0801)↓(0803)↓(E)↓(H)↓(A).	3) (080E)↓(0A02)↓(E)↓(B). ↑_____→

Продовження таблиці 12

В - 39 1) (0809) ⇔(0810); 2) (080A)↓(0903)↓(D)↓(H); 3) (0800)↓(E)↓(H)↓(A)↓(0901).	**В - 40** 1) (0803) ⇔(0808); 2) (0801)↓(D)↓(A)↓(0806)↓(E); 3) (0800)↓(0801)↓(0901)↓(H). ↑_____→
В - 41 1) (0902) ⇔(0901); 2) (0901)↓(B)↓(D)↓(A); 3) (0902)↓(H)↓(0901)↓(A). ↑_____→	**В - 42** 1) (0800) ⇔(0905); 2) (0901)↓(0801)↓(B)↓(H); 3) (0801)↓(B)↓(D)↓(A)↓(0802).
В - 43 1) (0809) ⇔(080A); 2) (0806)↓(0805)↓(H)↓(D)↓(A); 3) (0905)↓(0801)↓(D)↓(0803). ↑_____→	**В - 44** 1) (090C) ⇔(0915); 2) (0806)↓(D)↓(0807)↓(A)↓(0801); 3) (090A)↓(0801)↓(0802)↓(H). ↑_____→
В - 45 1) (0801) ⇔(090D); 2) (0806)↓(B)↓(D)↓(H); 3) (0905)↓(0902)↓(A)↓(0903). ↑_____→	**В - 46** 1) (080B) ⇔(0801); 2) (0815)↓(D)↓(B)↓(0806)↓(C); 3) (0907)↓(0806)↓(E)↓(A)↓(0802).

Продовження таблиці 12

В - 47	**В - 48**
1) (0903) ⇔(0803); 2) (B)↓(0901)↓(D)↓(0800); 3) (0801)↓(B)↓(H)↓(0802)↓(D).	1) (080C) ⇔(080A); 2) (0800)↓(D)↓(0801)↓(H); 3) (090A)↓(0803)↓(H)↓(0801). ↑_____→
В - 49	**В - 50**
1) (0805) ⇔(090A); 2) (090C)↓(B)↓(0901)↓(H)↓(A); 3) (0900)↓(D)↓(A)↓(H)↓(0901).	1) (080B) ⇔(0901); 2) (0900)↓(0903)↓(B)↓(D)↓(A); 3) (0907)↓(0905)↓(H)↓(0906). ↑_____→
В - 51	**В - 52**
1) (090B) ⇔(0801); 2) (0804)↓(0905)↓(090C)↓(H); 3) (0801)↓(D)↓(C)↓(A)↓(0807). ↑_____→	1) (0800) ⇔(0B00); 2) (080B)↓(C)↓(D)↓(0803); 3) (0905)↓(H)↓(0908)↓(A)↓(B). ↑_____→
В - 53	**В - 54**
1) (080E) ⇔(0802); 2) (0801)↓(0920)↓(080C)↓(A); 3) (B)↓(0801)↓(C)↓(A)↓(0807). ↑_____→	1) (0903) ⇔(0811); 2) (0805)↓(0A00)↓(0901)↓(H); 3) (0804)↓(B)↓(E)↓(A)↓(0903). ↑_____→

Продовження таблиці 12

В - 55	**В - 56**
1) (092A) ⇔(0800); 2) (0819)↓(A)↓(D)↓(080A); 3) (093A)↓(B)↓(0909)↓(A)↓(H). ↑_____→	1) (0909) ⇔(0B0A); 2) (0807)↓(0908)↓(D)↓(0802); 3) (0805)↓(H)↓(0808)↓(C)↓(B). ↑_____→
В - 57	**В - 58**
1) (0808) ⇔(083A); 2) (0806)↓(0902)↓(0900)↓(0802); 3) (0804)↓(E)↓(C)↓(A)↓(0801). ↑_____→	1) (092B) ⇔(0801); 2) (0800)↓(090A)↓(E)↓(0805); 3) (0901)↓(B)↓(0900)↓(C)↓(0800). ↑_____→

СПИСОК ЛІТЕРАТУРИ

1 Мікропроцесори. Терміни та визначення. ДСТУ 2397-94. – К.: Держстандарт України, 1994. – 11 с.

2 Методические указания по лабораторным работам по дисциплине “Микропроцессорные информационно-управляющие системы на железнодорожном транспорте”. – Харьков: ХИИТ, 1992. – Ч.2. – 48 с.
3 Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем: Учеб. для вузов. – С.Пб.: Питер, 2007. – 668 с.
4 Гилмор Ч. Введение в микропроцессорную технику. – М.: Мир, 1984. – 334 с.
5 Микропроцессоры и микропроцессорные комплекты интегральных микросхем: Справочник: в 2 т./ Н.Н. Аверьянов, А.И. Березенко, Ю.И. Борщенко и др.; Под ред. В.А. Шахнова. – М.: Радио и связь, 1988. – Т. 2. – 368 с.
6 Степанов А.Н. Архитектура вычислительных систем и компьютерных сетей. – С.Пб.: Питер, 2007. – 509 с.
7 Бройдо В.Л., Ильина О.П. Архитектура ЭВМ и систем: Учеб. для вузов. – С.Пб.: Питер, 2006. – 718 с.