НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

Тернопільський національний технічний університет імені Івана Пулюя

Кафедра автоматизації технологічних процесів і виробництв

Електроніка і мікропроцесорна техніка

Методичні вказівки для лабораторної роботи № 27

Реалізація керуючих дій в МП КР580ВМ80А

Тернопіль 2016

Методичні вказівки для лабораторної роботи №27. "Реалізація керуючих дій в МП КР580ВМ80А" з курсу "Електроніка і мікропроцесорна техніка". /Медвідь В.Р., Микулик П.М., Пісьціо В.П., Тернопіль: ТНТУ, 2016 - 11 с.

Для студентів напряму: 6.050202 "Автоматизоване управління технологічними процесами.

Методічні вказівки розглянуті і затверджені на засіданні кафедри автоматизації технологічних процесів і виробництв (протокол № 6 від 23.11.2015 року).

ЛАБОРАТОРНА РОБОТА № 27
РЕАЛІЗАЦІЯ КЕРУЮЧИХ ДІЙ В МП КР580ВМ80

Мета роботи

1. Вивчення принципів опитування зовнішніх пристроїв і способів організації роботи мікропроцесора в режимі очікування події.
2. Розгляд принципів формування сигналів і організації часової затримки.

Короткі теоретичні відомості

При розробці програмного забезпечення для мікропроцесорних систем, що використовуються для управління різними механізмами і машинами, а також призначених для реалізації на їх основі вбудованих систем (для регулювання параметрів технологічних процесів і виробництв) виникає необхідність програмування таких типових задач як:

- опитування стану двійкового датчика,
- очікування події,
- формування вихідних керуючих сигналів,
- формування часових затримок і т.п.

Розглянемо деякі способи програмної реалізації типових задач управління на базі однокристального восьмирозрядного мікропроцесора КР580, а також методику програмування обчислювальних процедур, розробка яких проблематична через обмеження, пов'язаних з особливостями організації самого мікропроцесора.

Мікропроцесорні системи управління технологічними об'єктами повинні враховувати події, які відбуваються в самому об'єкті, реагувати на них і виробляти керуючі впливи в так званому реальному масштабі часу.

Для початку розглянемо питання організації взаємодії системи із зовнішніми пристроями.

1. Опитування двійкового датчика

Взаємодія мікропроцесора з іншими елементами системи, а також із зовнішніми пристроями, здійснюється за допомогою спеціалізованих мікросхем, наприклад: КР580ВВ55, КР580ВН59, КР580ВТ57, КР580ВИ53, КР580ВВ51, КР580ВН28 і т.д.

Будемо вважати, що є підключення до мікропроцесора зовнішнього пристрою за допомогою багаторежимного буферного регістру (МБР) К589ІР12.

Спрощена схема підключення представлена на рис. 6 для випадку організації режиму вводу.

На рис. 1 є наступні позначення:

«CPU» - мікропроцесор КР580;

«D1» - багаторежимний буферний регістр К589ІР12;

«D2.1», «D2.2», ..., «D2.8» - мікросхеми, призначені для підсилення сигналу і гальванічної розв'язки;

«R1», «R2», ..., «R16» - опори;

«S1», «S2», ..., «S8» - ключі;

«VD1», «VD2», ..., «VD8» - світлодіоди;

«ША», «ШД» - шини адреси і даних;

«На ШУ» - показує напрямок передачі сигналу на шину управління системи;

«У Сх.ДщА» - показує напрямок передачі сигналу на дешифратор адреси;

«CS1», «CS2» - входи вибору кристалу;

«INT» - вхід запиту переривання;

«R» - вхід для встановлення нуля;

«STB» - вхід для стробуючого сигналу;

«MD» - вхід вибору режиму;

Додатково на схемі так само подано лінії живлення «+5» і загальна лінія.

У схемі ключі імітують поведінку дискретного контактного датчика типу «сухий контакт». Якщо відповідний ключ розімкнути, то на відповідному вході D присутній високий рівень напруги (логічна одиниця), якщо відповідний ключ замкнутий - то на однойменному вході маємо логічний нуль.

Світлодіоди в схемі призначені для індикації наявного числа на порту.

Будемо вважати, що регістр D1 підключений до порту «01» мікропроцесора.

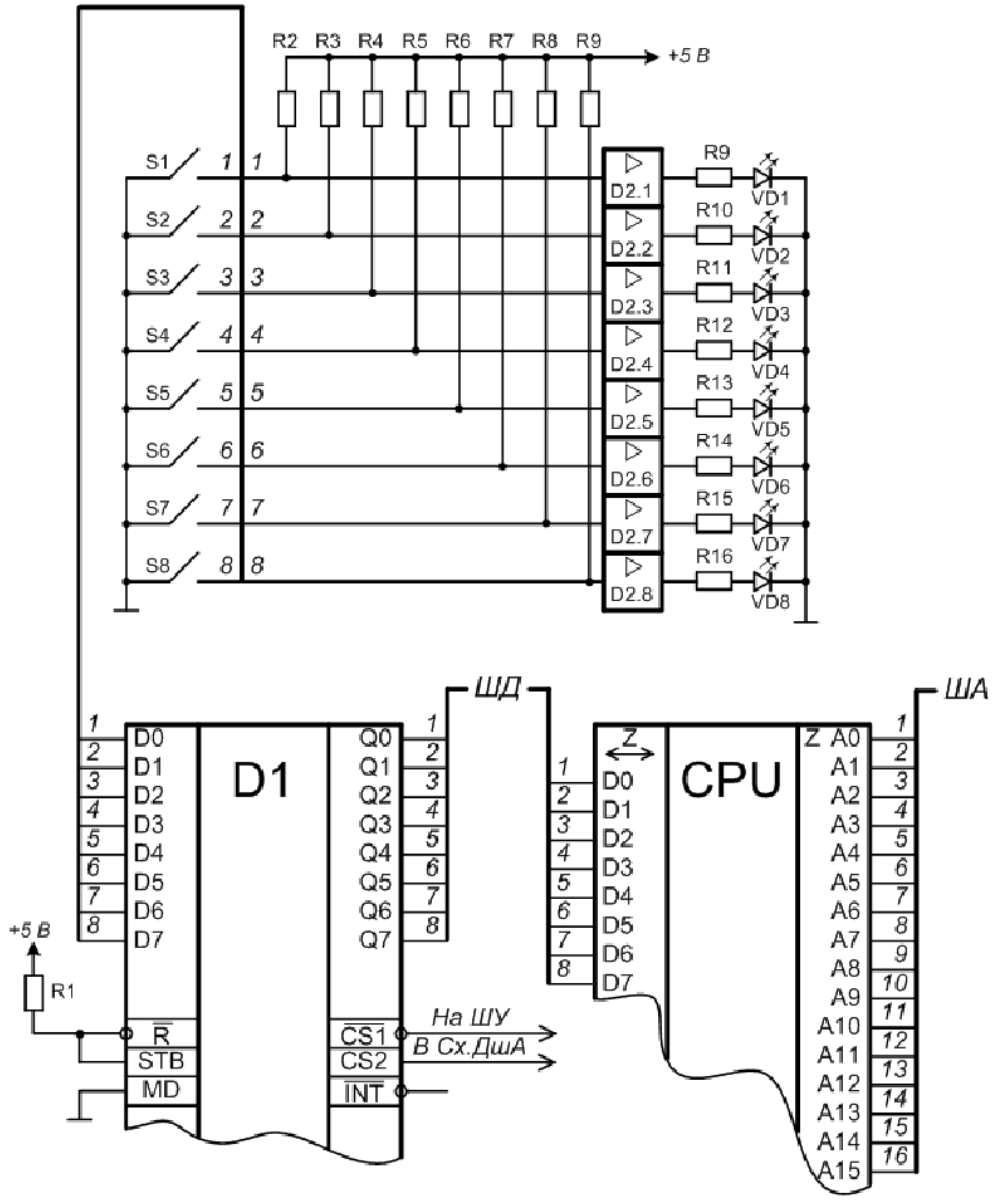

Рис. 1 Схема організації вводу за допомогою К589ИР12

Нехай, наприклад, потрібно в деякій частині керуючої програми опитати сигнал від четвертого датчика (ключ «S4», лінія порту «D3») і в залежності від результату опитування передати управління фрагменту програми з міткою LABEL_N, якщо датчик не спрацював (тобто D3 = 1) або за адресою зазначеного міткою LABEL_Y, якщо датчик спрацював (тобто D3 = 0).

На рис. 2 наводиться алгоритм Програми 3, який реалізує описану вище процедуру опитування.

Програма має символічне ім'я INPKEY (ввід ключа), яке використовується в якості мітки першої команди цієї програми.

Велика частина етапів алгоритму (рис. 2) вже зустрічалася раніше і їх програмна реалізація не складе труднощів.

Зупинимося детальніше на процедурі виділення певного біту.

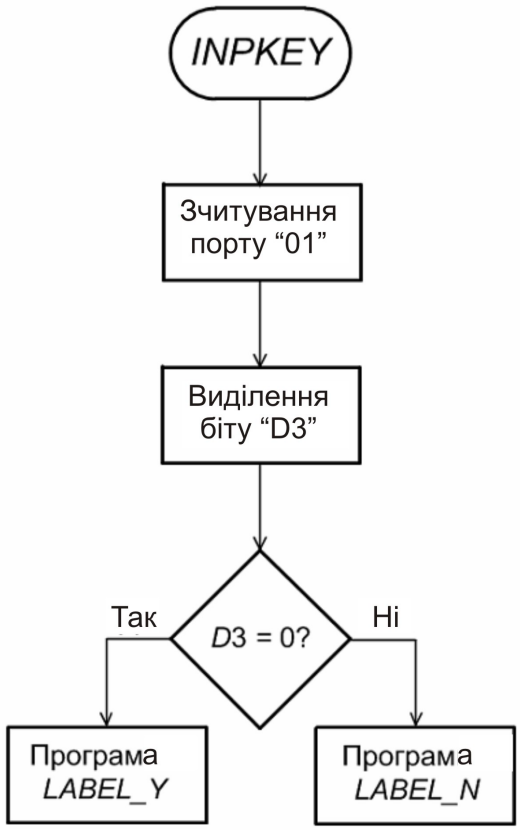

Рис. 2 Алгоритм Програми 3

У багатьох випадках при виконанні програм необхідно перевіряти, чи змінювався стан одного або декількох розрядів числа, що зберігається в акумуляторі.

Така процедура отримала найменування «маскування», тому в цій процедурі, крім вмісту акумулятора, використовується заздалегідь задане число, або по-іншому - «маска».

В масці значення кожного з бітів задаються, виходячи з бажаного результату.

Сама процедура маскування полягає в застосуванні однієї з логічних операцій над вмістом акумулятора і маски, що дозволяє потім використовувати команди умовних переходів по флагах «S», «Z», «P».

Якщо задати маску так, що б у ній логічні одиниці «1» були в тих розрядах, які слід залишити незмінними, а логічні нулі «0» поставити в розрядах, що підлягають очистці, то в результаті логічного множення (операція «I») акумулятора і маски можна здійснити виділення потрібних розрядів.

При цьому, якщо логічна «1» була тільки в одному розряді маски, то в результаті можна здійсни-

ти наступну перевірку:

- якщо результат дорівнює «0» - то і відповідний біт (на його позиції в масці була «1») дорівнював «0»,
- в іншому випадку цей біт дорівнював «1».

Якщо задати маску так, щоб в ній логічні одиниці «1» були в тих розрядах, які слід примусово встановити в одиницю «1», а логічні нулі «0» поставити в розрядах, що не підлягають зміні, то в результаті логічного додавання (операція «АБО») акумулятора і маски можна здійснити примусову установку потрібних бітів в одиниці «1».

Якщо задати маску так, що б у ній логічні одиниці «1» були на тих позиціях, які слід інвертувати, і логічні нулі «0» - на тих позиціях, які варто залишити без зміни, то в результаті застосування операції «виключаюче АБО» над вмістом акумулятора і маски можна отримати інверсію зазначених в масці бітів (в цих позиціях стоять логічні одиниці «1»).

Додатково варто звернути увагу на ще одну важливу властивість операції «виключаюче АБО». Якщо задана маска і число в акумуляторі побітово рівні, то після виконання даної операції результат буде дорівнювати числу, у якого в усіх бітах стоять логічні нулі «0» (тобто воно арифметично дорівнює «0»).

Схожа властивість є у операції «виключаюче АБО в інверсії »(якщо така команда присутня в системі команд МП): якщо маска є побітовою інверсію числа в акумуляторі, то в результаті виконання даної операції вийде результат, у якого в усіх бітах стоять логічні нулі «0» (тобто він арифметично дорівнює «0»).

Приклади використання вищеописаних команд наведені в табл. 1.

Особливо підкреслимо, що процедура маскування з використанням логічних команд не забезпечує можливість переходу по флагу «С» (він встановлюється в «0») і «АС» - який

взагалі не змінюється. При цьому є можливість проведення маскування в разі наявності маски в одному з регістрів.

З урахуванням усього вищесказаного алгоритм Програми 3, представлений на рис. 2 може бути записаний у вигляді програми на асемблері (табл. 2).

Таблиця 1

Приклади використання логічних команд для маскування

Мнемокод	Машинний код	Число в акумуляторі	Маска, d8	Коментар	Результат в акумуляторі
ANI d8	E6 d8	0011 1010 1111 1111 0000 0000 1010 1010 1111 0000	1010 1100 0010 0010 0010 0010 0010 0010 1111 1111	Логічне множення вмісту акумулятора з d8	0010 1000 0010 0010 0000 0000 0010 0010 1111 0000
ORI d8	F6 d8	0011 1010 0000 1111 1111 0000	1010 1100 0000 1111 0000 1111	Логічне додавання вмісту акумулятора з d8	1011 1110 0000 1111 1111 1111
XRI d8	EE d8	0011 1010 0000 1111 1111 0000	1010 1100 0000 1111 0000 1111	Логічне “виключаюче АБО” вмісту акумулятора з d8	1001 0110 0000 0000 1111 1111

Таблиця 2

Програма 3

№ етапу	Коментар по етапу алгоритму	Мітка	Мнемокод	Коментар до команди
1	Зчитування з порту з адресою “01*h*”	INPKEY:	*IN 01h*	Призначення команди *IN*: Дані з порту завантажуються в “A”
2	Виділення біту “d3”, маска “0000 0100*b*” (”04*h*”)		*ANI 04h*	Призначення команди *ANI*: Над вмістом “A” та другого байту виконується операція “логічне множення”. Рзультат завантажується в “A”.
3	Перевірка біту “d3”=0 (спрацював?), Якщо “так”, то на “*Label_N*”; інакше далі на “*Label_Y*”		*JZ LABEL_Y*	Призначення команди *JZ*: Якщо вміст “A”=0, то відбувється перехід за адресою, що вказується у другому і третьому байтах команди. Якщо умова не виконується, то виконується наступна команда програми.
	Підпрограма “*Label_N*”:	LABEL_N:	…	…
	…	…	…	…
	Підрпрограма “*Label_Y*”:	LABEL_Y:	…	…

3. Формування керуючого сигналу

Якщо взаємодія мікропроцесорної системи з зовнішнім середовищем забезпечена, то очевидно, що наступним етапом створення повноцінної мікроконтролерної автоматичної системи регулювання є опрацювання питань вироблення керуючого сигналу.

Аналогічно пункту 1 будемо вважати, що маємо простий випадок підключення до мікропроцесора зовнішнього пристрою, наприклад, багаторежимного буферного регістру (МБР) К589ІР12.

Спрощена схема підключення представлена на рис. 3 для випадку організації режиму виводу.

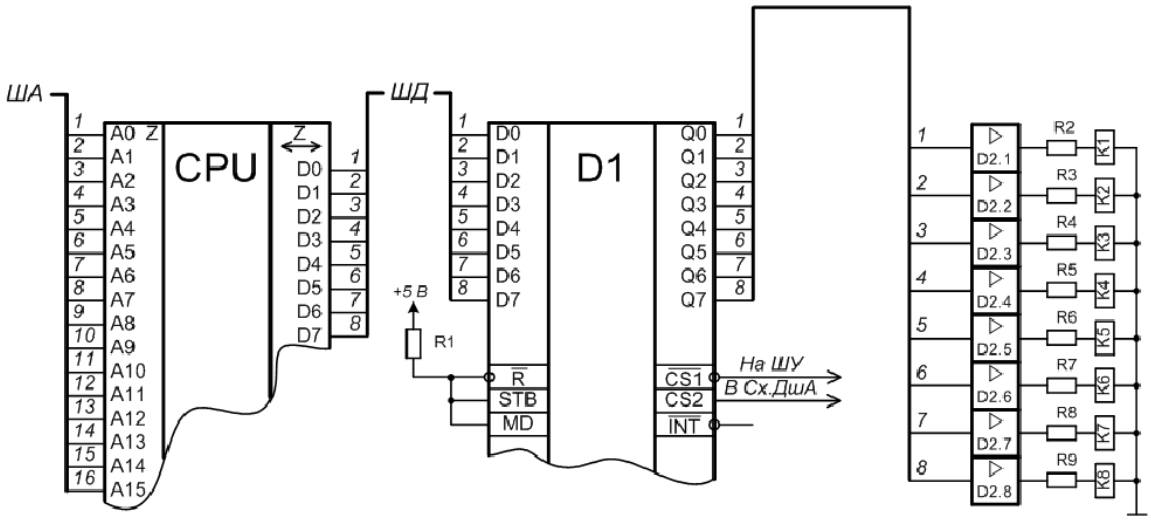

Рис. 3 Схема організації виводу за допомогою К589ИР12

У схемі рис. 3 «К1», «К2», ..., «До 8» це слабкострумові (по вхідних струмах) реле, які комутують, наприклад, ланцюги живлення виконавчого механізму, що працює за принципом «включено» / «вимкнено», і який відкриває або закриває ключ. При цьому «К1» може відповідати за рух виконавчого механізму №1 в один бік (умовно назвемо її «більше»), а «К2» - відповідати за рух цього ж виконавчого механізму №1 в інший бік (умовно назвемо її «менше»).

Можлива й інша логіка підключення, можлива також організація управління декількома механізмами, при цьому очевидно, що на порт необхідно подавати цілий байт (8 біт) так званого керуючого слова.

Подання керуючого впливу здійснюється за допомогою команди «OUT N», де «N» - номер порту виводу, а керуюче слово повинно зберігатися в акумуляторі «А».

4. Часова затримка

Якщо при виконанні робочої програми мікропроцесору потрібно функціонувати в реальному масштабі часу, наприклад, проводити опитування порту не регулярно, а через заданий час; або формувати керуючий вплив певної тривалості; або чекати настання події протягом встановленого часового інтервалу і т.п., то очевидно, що необхідний механізм підрахунку реально витраченого часу в секундах.

Програмна реалізація часової затримки використовує метод організації циклів. При цьому в деякий робочий регістр блоку РОН завантажується число, яке потім в кожному проході циклу зменшується на «1». Так триває до тих пір, поки вміст робочого регістра не стане рівним «0», що інтерпретується програмою як момент виходу з програмного циклу.

Отриманий час затримки (протягом цього часу процесор не виконував робочу програму) при цьому визначається числом, завантаженим в робочий регістр, і часом виконання команд, що утворюють програмний цикл.

Схема алгоритму часової затримки (Програма 4) показана на рис. 4. Вхід в алгоритм має найменування TIME і може використовуватися як посилання при організації складних

програм, якщо використовувати структуру типу «підпрограма».

Фрагмент програмного коду, отриманого згідно алгоритму на рис. 4, представлений в табл. 3.

Для отримання необхідної величини часової затримки необхідно визначити значення числа «X», що завантажується в робочий регістр «B». Число «X» визначається на основі розрахунку часу виконання команд, що утворюють тіло циклу.

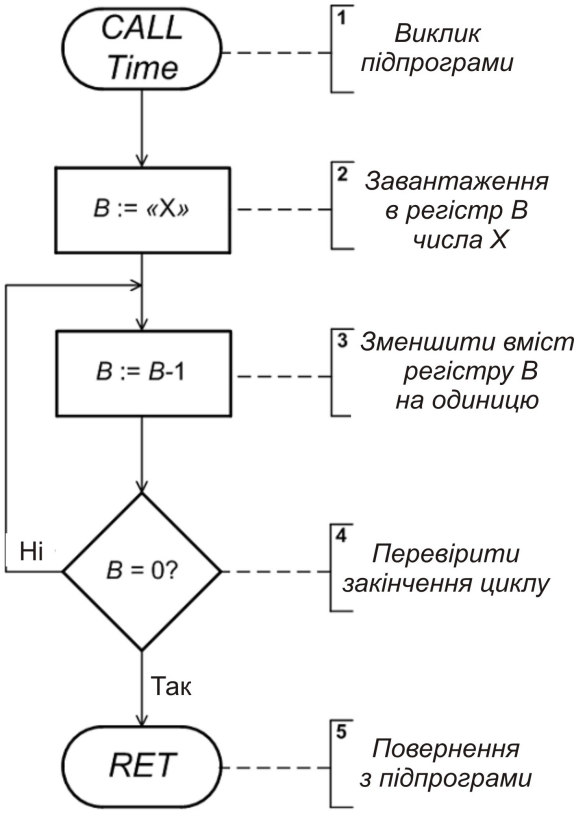

Рис. 4 Часова затримка

Припустимо, що в програмі, керуючої роботою мікропроцесорної системи, МП працює з тактовою частотою 2 МГц (період становить 0,5 мкс).

Необхідно реалізувати часову затримку тривалістю 251 мкс.

В описі системи команд КР580 вказується, за скільки тактів основної частоти синхронізації виконується кожна команда.

На основі цих даних можна записати:
CALL TIME - 17 тактів = 8,5 мкс;
MVI B, «X» - 7 тактів = 3,5 мкс;
DCR B - 5 тактів = 2,5 мкс;
JNZ Mk - 10 тактів = 5 мкс;
RET - 11 тактів - 5,5 мкс.

Таким чином, одноразово виконувані команди «CALL», «MVI», «RET» в цій підпрограмі вимагають 17,5 мкс (8,5 + 3,5 + 5,5).

Отже, для отримання необхідної затримки в 251 мкс, необхідно виконати команди «DCR» і «JNZ» стільки разів, щоб час їх виконання склав 233,5 мкс (251 - 17,5):

$$X = (233,5) / (2,5 + 5,0) \approx 31,$$

при цьому загальна затримка складе (31 * 7,5 + 17,5) = 250 мкс.

Якщо точність програмної реалізації часової затримки тривалістю 251 мкс з похибкою 1 мкс задовольняє розробника, то на цьому процедура вирішення зупиняється.

В іншому випадку необхідно скоригувати програму, додавши незначні операції (як в тіло циклу, так і поза нього) і перерахувавши загальний час виконання програми.

Реалізація часової **затримки більшої тривалості** при відносно низькій частоті синхронізації 2 МГц, з використанням описаного вище методу є неможливою, тому що максимального вмісту регістру мікропроцесора, що складає «FFh», не вистачить для того, щоб представити число X, для формування затримки, рівної, наприклад 1 сек.

Сформувати таку велику (для МП) затримку можна з використанням методу вкладених циклів подібно до того, як це показано на рис. 5 (алгоритм для Програми 5) і в таблиці 4 (код Програми 5).

Таблиця 3

Программа 4. «Часова затримка»

№ етапу	Коментар по етапу алгоритму	Мітка	Мнемокод	Коментар до команди
1	Виклик підпрограми “Time”		`CALL TIME`	Призначення команди *CALL*: [SP – 1] ← PCH; [SP – 2] ← PCL; SP ← (SP – 2); PC ← adr; Старші 8 бітів адреси наступної команди пересилаються в комірку пам’яті, адреса якої на одиницю менша за вміст вказівника стеку *SP*. Молодші 8 бітів адреси наступної команди пересилаються в комірку пам’яті, адреса якої на два менша за вміст вказівника стеку *SP*. Вміст вказівника стеку зменшується на 2. Управління передається команді, адреса якої вказана у другому та третьому байтах команди виклику підпрограми
…	…	…	…	…
2	Завантаження в “B” числа “X”	TIME:	`MVI B,«X»`	Призначення команди *MVI*: Другий байт команди “X” пересилається в регістр “B”
3	Зменшити “B” на одиницю	Mk:	`DCR B`	Призначення команди *DCR*: Вміст регістру “B” зменшується на одиницю
4	Перевірити закінчення циклу		`JNZ Mk`	Призначення команди *JNZ*: Якщо останній результат не дорівнює “0”, то перехід за міткою Mk, яка є явно заданою адресою частини програми
5	Повернення з підпрограми		`RET`	Призначення команди *RET*: PCL ← [SP]; PCH ← [SP + 1]; SP ← (SP + 2); Вміст комірки пам’яті, адреса якої знаходиться в *SP*, пересилається в молодші 8 бітів програмного лічильника PC. Вміст комірки пам’яті, ядреса якої на 1 більша за вміст *SP*, пересилається в старші 8 бітів PC. Вміст *SP* збільшується на два. Управління повертається команді, наступній після *CALL*

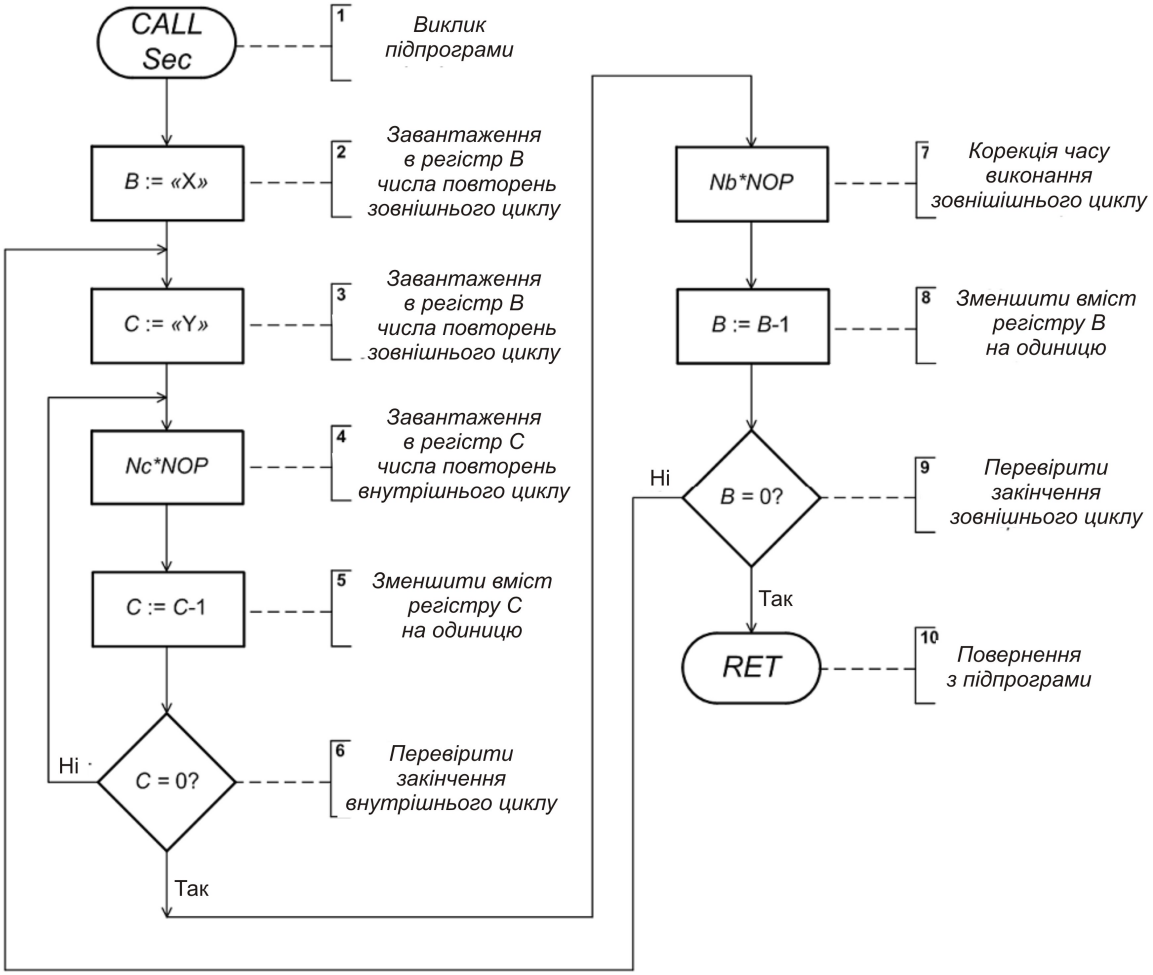

Рис. 5 Часова затримка

Таблиця 4

Программа 5. «Часова затримка»

№ етапу	Коментар по етапу алгоритму	Мітка	Мнемокод	Коментар до команди
1	Виклик підпрограми “Sec”		`CALL Sec`	Призначення команди *CALL*: [SP – 1] ← PCH; [SP – 2] ← PCL; SP ← (SP – 2); PC ← adr; Управління передвється команді, адреса якої вказана у другому та третьому байтах команди
…	…	…	…	…
2	Завантаження в “B” кількості повторень зовнішнього циклу	Sec:	`MVI B,FBh`	Призначення команди *MVI*: Другий байт команди *“Fbh”* пересилається в регістр “B”
3	Завантаження в “C” кількості повторень внутрішнього циклу	Mk1:	`MVI C,E3h`	Призначення команди *MVI*: Другий байт команди *“Ebh”* пересилається в регістр “*C*”
4	Корекція часу виконання внутрішнього циклу	Mk2:	`NOP` `NOP` `NOP` `NOP` `NOP`	Призначення команди *NOP*: Немає операції

5	Зменшити лічильник внутрішнього циклу на одиницю		`DCR C`	Призначення команди *DCR*: Вміст регістра “C” зменшується на одиницю
6	Перевірити закінчення циклу		`JNZ Mk2`	Призначення команди *JNZ*: Якщо останній результат не дорівнює “0”, то перехід за міткою Mk2, яка є явно заданою адресою частини програми
7	Корекція часу виконання зовнішнього циклу		`NOP`	Призначення команди *NOP*: Немає операції
8	Зменшити лічильник зовнішнього циклу на одиницю		`DCR B`	Призначення команди *DCR*: Вміст регістра “B” зменшується на одиницю
9	Перевірити закінчення циклу		`JNZ Mk1`	Призначення команди *JNZ*: Якщо останній результат не дорівнює “0”, то перехід за міткою Mk1, яка є явно заданою адресою частини програми
10	Повернення з підпрограми		`RET`	Призначення команди *RET*: PCL ← [SP]; PCH ← [SP + 1]; SP ← (SP + 2); Управління повертається команді, наступній після *CALL*

Завдання 1. Аналіз роботи мікропроцесора в режимі очікування події

Скласти програму очікування події, перевести її в машинний код, записати в ОЗП емулятора і перевірити правильність роботи.

Як «події» (які потрібне очікувати) прийняти факт замикання будь-якого із зазначених в табл. 6 ключів (датчиків, позначених «X»), підключених до порту «05h».

Таблиця 6

Номери ключів, спрацьовування яких необхідно виявити

№ ключа	№ біту порта	Варіанти														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1	D0	x	x		x	x			x	x				x		
S2	D1			x		x		x		x				x	x	
S3	D2			x				x		x			x	x	x	
S4	D3			x				x		x			x		x	
S5	D4			x			x	x	x	x	x		x	x	x	x
S6	D5			x			x				x		x		x	
S7	D6			x	x		x				x		x		x	
S8	D7	x		x	x		x					x	x		x	

Примітка: в разі відсутності в використовуваному емуляторі можливості зміни стану на обраній зовнішній лінії (шляхом зміни положення ключів або перемикачів), необхідно змінити програму, і замість команди «IN N» використовувати звернення до групи послідовно розташованих комірок пам'яті (очікувати факт виявлення в розрядах чергової комірки пам'яті необхідної умови).

Оброблювані комірки розмістити, починаючи з адреси «0B00h», і заповнити на свій розсуд.

Необхідно при цьому передбачити можливість визначення, на якій адресі відбулася подія.

Завдання 2. Дослідження роботи мікропроцесора в режимі формування керуючого сигналу

Створити програму з формування керуючого впливу, що подається на групу виконавчих механізмів, які через систему гальванічної розв'язки підключені до порту «07h».

Програму перевести в машинний код, записати в пам'ять емулятора і досліджувати в покроковому режимі.

Інформація про те, який механізм потрібно включити, представлена для кожного з варіантів у вигляді табл. 7. Не зазначені в таблиці механізми потрібно вимкнути.

Таблиця 7

Стан виконавчих механізмів

№ механізму	№ біту порту	Варіанти														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M1	D0	вкл.	вкл.	вкл.	вкл.	вкл.	вкл.			вкл.		вкл.	вкл.		вкл.	
M2	D1		вкл.					вкл.	вкл.	вкл.		вкл.		вкл.		
M3	D2	вкл.			вкл.	вкл.	вкл.	вкл.	вкл.		вкл.			вкл.	вкл.	вкл.
M4	D3			вкл.	вкл.						вкл.		вкл.	вкл.	вкл.	вкл.
M5	D4			вкл.		вкл.			вкл.	вкл.	вкл.	вкл.	вкл.			вкл.
M6	D5	вкл.	вкл.		вкл.		вкл.	вкл.	вкл.	вкл.		вкл.	вкл.		вкл.	вкл.
M7	D6	вкл.	вкл.		вкл.	вкл.	вкл.		вкл.	вкл.				вкл.		
M8	D7		вкл.	вкл.		вкл.		вкл.				вкл.	вкл.	вкл.		вкл.

Завдання 3. Створення програми формування заданої часової затримки

Використовуючи інформацію про принципи і методи формування програмної затримки необхідно:

- розрахувати точно час виконання Програми 5;
- модифікувати програму так, що б час затримки становило величину, зазначену в табл. 8;
- перевести отриману програму в машинний код, записати в емулятор і досліджувати її виконання.

Таблиця 8

Необхідний час програмної затримки

Час в сек. для вказаного варіанту														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0,3	*1,25*	*1,4*	*0,75*	*0,4*	*0,67*	*1,3*	*2,0*	*0,12*	*0,66*	*0,9*	*1,6*	*2,5*	*1,5*	*0,57*

Примітка: необхідно врахувати, що час «емуляції» при запуску досліджуваної програми може не збігтися з розрахунковим часом роботи процесора, тому що емулятор працює з частотою, більшою, ніж 2 МГц.

Контрольні питання

1. Яким чином в мікропроцесорної системі може відбуватися опитування дискретного датчика?
2. Як можна виділити певний розряд в двійковому числі?

3. Яким чином може бути використана команда логічного множення при маскуванні даних?
4. Як використовується роз'єднання при маскуванні?
5. Для чого може бути використана логічна операція виключаюче «АБО» (виключаюче «АБО» в інверсії)?
6. У чому суть роботи системи в режимі очікування події?
7. Як можна сформувати керуючий сигнал в мікропроцесорній системі управління?
8. Наведіть програмні способи формування часової затримки.
9. Де може бути використана часова затримка?

Список рекомендованої літератури

1. Самофалов К. Г., Викторов О. В. Микропроцессоры. – Б-ка инеженера – 2-е изд., перераб. и доп. - К: Техника, 1989.-312 с.
2. Шевкопляс Б.В. Микропроцессорные структуры. Инженерные решения: Справочник. 2-е изд., перераб. и доп. -М.: Радио и связь, 1990, -512с.
3. Угрюмов Е.П. Цифровая схемотехника. Спб.: BHV, 2001. 528 с.
4. Злобин В. К-, Григорьев В. Л. 58 Программирование арифметических операций в микропроцессорах: Учеб. пособие для технических вузов.— М.: Высш. шк., 1991. —303 с.: ил.