НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

Тернопільський національний технічний університет імені Івана Пулюя

Кафедра автоматизації технологічних процесів і виробництв

Лабораторна робота № 6 з курсу ”Мікропроцесорні та програмні засоби автоматизації”

Вивід даних на 7- сегментний дисплей на програмному симуляторі PIC Simulator IDE

Тернопіль 2018

Методичні вказівки до лабораторної роботи №6 «Вивід даних на 7-сегментний дисплей на програмному симуляторі PIC Simulator IDE» з курсу «Мікропроцесорні та програмні засоби автоматизації». Медвідь В.Р., Пісьціо В.П., - Тернопіль: ТНТУ, 2018 - 17 с.

Відповідальні за випуск доцент, к.т.н. Медвідь В.Р.,
асистент Пісьціо В.П.

Для студентів напряму: 151 "Автоматизація та комп'ютерно-інтегровані технології"

Лабораторна робота №6

Вивід даних на 7- сегментний дисплей на програмному симуляторі PIC Simulator IDE

1. Послідовність роботи з програмним симулятором PIC Simulator IDE

Основне вікно програми PIC Simulator IDE має вигляд, показаний на (рис. 1).

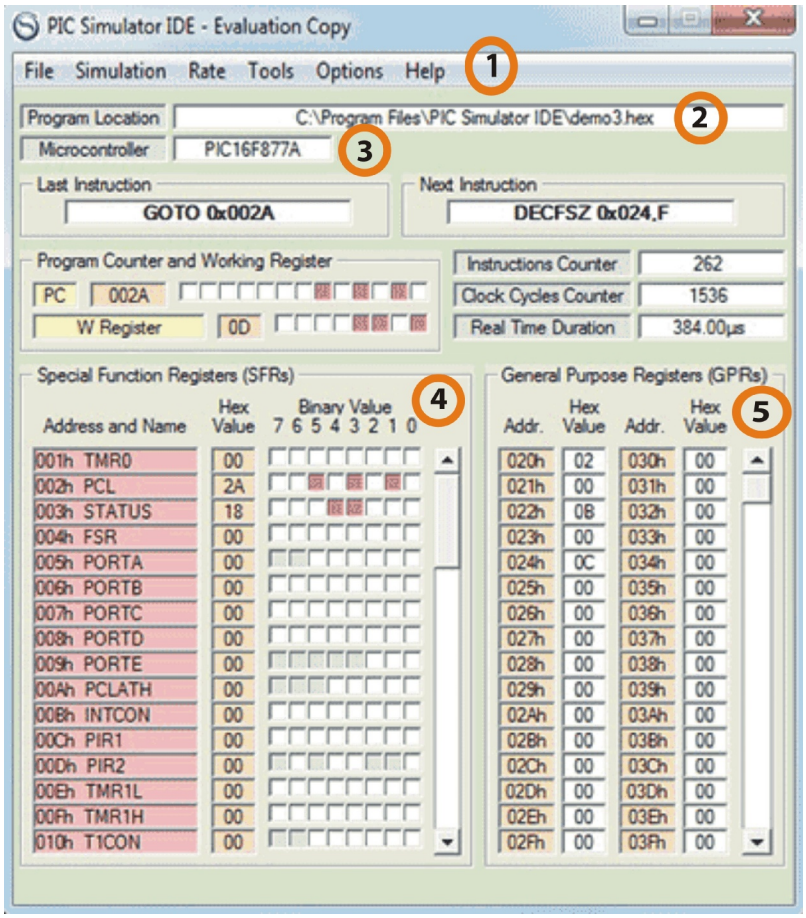

Рис. 1. Основне вікно програми PIC Simulator IDE

У верхній частині знаходяться меню, через які можна отримати доступ до основних і додаткових модулів програми (поз. 1)(рис. 1).

В рядку Program Location вказано шлях до обраної програми і її ім'я (поз. 2).

В рядку Microcontrollers, відображається тип обраного мікроконтролера (поз. 3).

У нижній частині вікна є дві панелі (поз.4 і поз.5), в яких відображаються стан програми, вміст регістрів спеціальних функцій (РСФ) і керуючих регістрів обраного МК.

Послідовність роботи з програмним симулятором наступний:

- запуск програми PIC Simulator IDE;
- вибір типу мікроконтролера, для якого написана програма;
- вибір частоти кварцового генератора (впливає тільки на відображувані програмою дані про час виконання програми або команди, але не на швидкість роботи програми, що налагоджуються в PIC Simulator IDE);
- завантаження програми у вигляді HEX-файлу або запуск вбудованого компілятора мови асемблера і написання в ньому потрібної програми;
- вибір потрібних модулів віртуальних пристроїв;
- вибір швидкості і режиму роботи програми симулятора;
- запуск процесу симуляції роботи програми на обраному МК.

Якщо потрібно скористатися для роботи з симулятором власною програмою або внести зміни у вже розроблену, необхідно створити або завантажити для цього файл асемблера, з якого після компіляції буде створений необхідний для роботи з симулятором hex-файл.

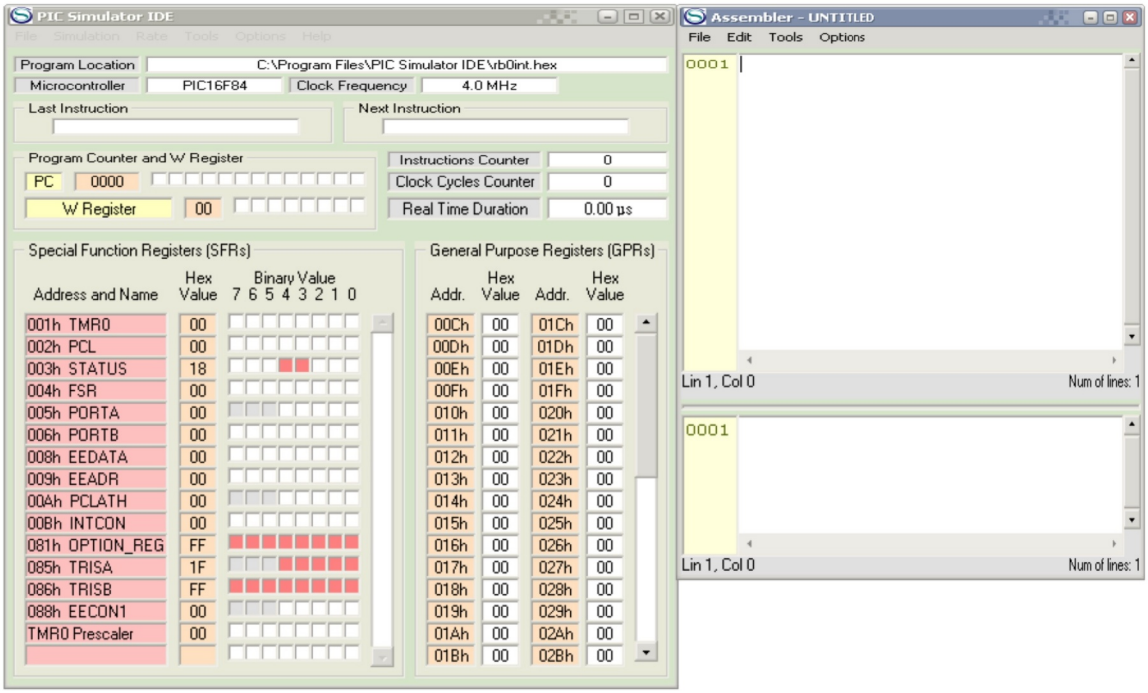

Рис. 2 Вікно симулятора з відкритим вікном Assembler

Для цього:

1. Натиснути Options | Assembler. Відкриється вікно компілятора Assembler – UNTITLED (рис. 2);

2. У вікні Assembler натиснути опцію File. Розкриється закладка (рис. 3), з якої для створення нового файлу потрібно натиснути New, а для завантаження вже створеного – OPEN.

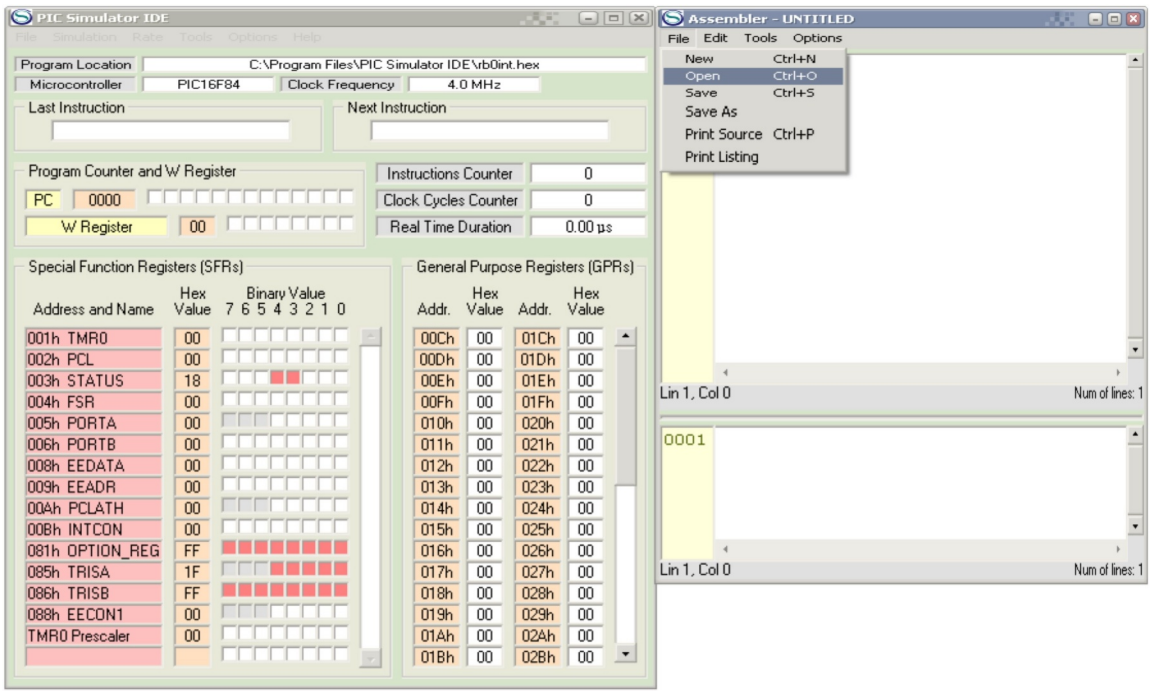

Рис. 3 Завантаження існуючого або створення нового файлу асемблера

3. Після вибору і завантаження файлу (наприклад, rb0int.asm), його текст з’явиться у вікні Assembler (рис. 4).

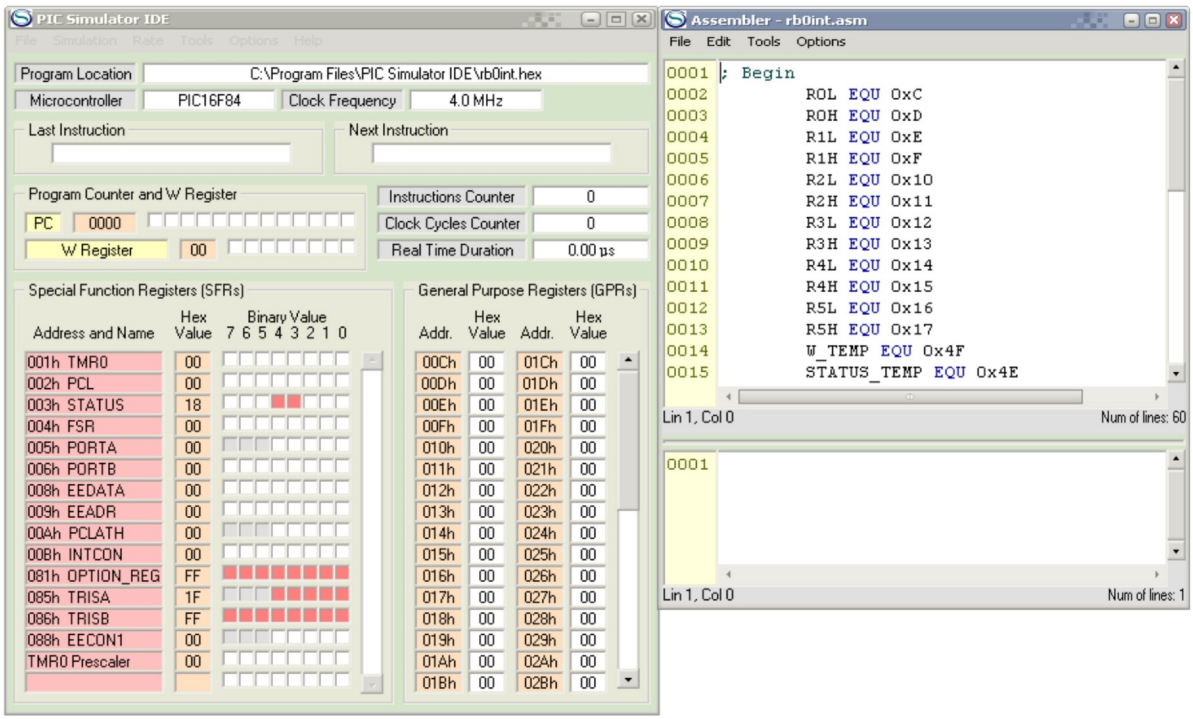

Рис. 4 Завантаження файлу rb0int.asm

4. Для компіляції створеного або завантаженого і потім зміненого файлу, натисніть Tools і у вікні, що розкриється – Assemble. В нижній половині вікна Assembler з’явиться лістинг відкомпільованого файлу і, одночасно, при відсутності помилок, буде створений одноіменний hex-файл.

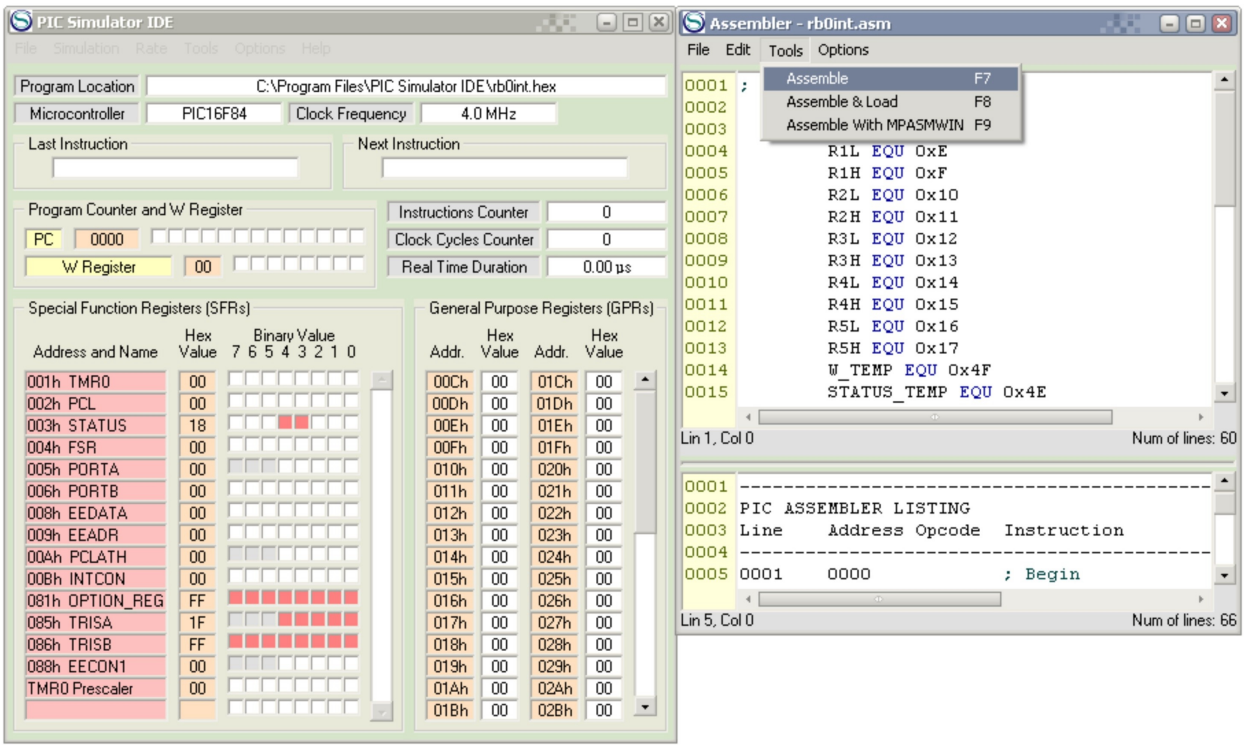

Рис. 5

2. **Завдання на лабораторну роботу:** ввід-вивід даних через порти PIC-контролера.

1. Вивчити програмну модель PIC Simulator IDE.

2. Вивчити команди арифметичних операцій PIC – контролера.

3. Дослідити роботу програми з Прикладу 1 та вміст регістрів контролера (W, STATUS…), які використовуються при виконанні цієї програми.

4. Записати для вибраних команд асемблера коментар щодо їх призначення (див. Приклад 2).

Приклад 1.

Програма відображення даних на 7- сегментному дисплеї. Ця програма відображає числа від 0 до 99 на двох 7- сегментних дисплеях, які підключені паралельно і управляються двома лініями. Для перемикання використовується процедура мультиплексування переривання TMR0.

Текст програми з файлу «7segment.asm» має наступний вигляд:

```
; Begin
        R0L EQU 0x20
        R0H EQU 0x21
        R1L EQU 0x22
        R1H EQU 0x23
        R2L EQU 0x24
        R2H EQU 0x25
        R3L EQU 0x26
        R3H EQU 0x27
        R4L EQU 0x28
        R4H EQU 0x29
        R5L EQU 0x2A
        R5H EQU 0x2B
        W_TEMP EQU 0x7F
        STATUS_TEMP EQU 0x7E
        ORG 0x0000
        BCF PCLATH,3
        BCF PCLATH,4
        GOTO L0005
        ORG 0x0004
        MOVWF W_TEMP
        SWAPF STATUS,W
        CLRF STATUS
        MOVWF STATUS_TEMP
        CALL L0006
        SWAPF STATUS_TEMP,W
        MOVWF STATUS
        SWAPF W_TEMP,F
        SWAPF W_TEMP,W
        RETFIE
L0005:
; 1: Dim digit As Byte 'input variable for GETMASK subroutine
;       The address of 'digit' is 0x2D
        digit EQU 0x2D
; 2: Dim digit1 As Byte 'current high digit
;       The address of 'digit1' is 0x2E
        digit1 EQU 0x2E
; 3: Dim digit2 As Byte 'current low digit
```

```
;       The address of 'digit2' is 0x2F
        digit2 EQU 0x2F
; 4: Dim mask As Byte 'output variable from GETMASK subroutine
;       The address of 'mask' is 0x30
        mask EQU 0x30
; 5: Dim mask1 As Byte 'current high digit mask
;       The address of 'mask1' is 0x31
        mask1 EQU 0x31
; 6: Dim mask2 As Byte 'current low digit mask
;       The address of 'mask2' is 0x32
        mask2 EQU 0x32
; 7: Dim i As Byte
;       The address of 'i' is 0x33
        i EQU 0x33
; 8: Dim phase As Bit
;       The address of 'phase' is 0x2C,0
; 9:
; 10: Symbol d1enable = PORTC.0 'enable line for higher 7-segment display
;       The address of 'd1enable' is 0x7,0
; 11: Symbol d2enable = PORTC.1 'enable line for lower 7-segment display
;       The address of 'd2enable' is 0x7,1
; 12: TRISB = %00000000 'set PORTB pins as outputs
        BSF STATUS,RP0
        CLRF 0x06
        BCF STATUS,RP0
; 13: TRISC.0 = 0 'set RC0 pin as output
        BSF STATUS,RP0
        BCF 0x07,0
        BCF STATUS,RP0
; 14: TRISC.1 = 0 'set RC1 pin as output
        BSF STATUS,RP0
        BCF 0x07,1
        BCF STATUS,RP0
; 15: d1enable = False
        BCF 0x07,0
; 16: d2enable = False
        BCF 0x07,1
; 17: mask1 = 0
        CLRF 0x31
; 18: mask2 = 0
        CLRF 0x32
; 19: phase = 0
        BCF 0x2C,0
; 20: INTCON.T0IE = 1 'enable Timer0 interrupts
        BSF 0x0B,5
; 21: INTCON.GIE = 1 'enable all un-masked interrupts
        BSF 0x0B,7
; 22: OPTION_REG.T0CS = 0 'set Timer0 clock source to internal instruction cycle clock
        BSF STATUS,RP0
        BCF 0x01,5
        BCF STATUS,RP0
; 23:
```

```
; 24: loop:
L0001:
; 25: For i = 0 To 99
        CLRF 0x33
L0007:
        MOVF 0x33,W
        SUBLW 0x63
        BTFSS STATUS,C
        GOTO L0008
; 26: digit1 = i / 10 'get current high digit
        MOVF 0x33,W
        MOVWF R0L
        CLRF R0H
        MOVLW 0x0A
        MOVWF R1L
        CLRF R1H
        CALL D001
        MOVWF 0x2E
; 27: digit2 = i Mod 10 'get current low digit
        MOVF 0x33,W
        MOVWF R0L
        CLRF R0H
        MOVLW 0x0A
        MOVWF R1L
        CLRF R1H
        CALL D001
        MOVF R2L,W
        MOVWF 0x2F
; 28: TMR0 = 0 'reset Timer0 to prevent its interrupt before both masks are determined
        CLRF 0x01
; 29: digit = digit1
        MOVF 0x2E,W
        MOVWF 0x2D
; 30: Gosub getmask 'get mask for high digit
        CALL L0002
; 31: mask1 = mask
        MOVF 0x30,W
        MOVWF 0x31
; 32: digit = digit2
        MOVF 0x2F,W
        MOVWF 0x2D
; 33: Gosub getmask 'get mask for low digit
        CALL L0002
; 34: mask2 = mask
        MOVF 0x30,W
        MOVWF 0x32
; 35: Gosub show1 'display new mask
        CALL L0003
; 36: Gosub show2 'display new mask
        CALL L0004
; 37: WaitUs 500 'delay interval suitable for simulation
        MOVLW 0xF4
```

```
        MOVWF R4L
        MOVLW 0x01
        MOVWF R4H
        CALL Y001
; 38:  'use large delay for the real device, say WAITMS 500
; 39: Next i
        MOVLW 0x01
        ADDWF 0x33,F
        BTFSS STATUS,C
        GOTO L0007
L0008:  MOVLW 0x1F
        ANDWF STATUS,F
; 40: Goto loop
        GOTO L0001
; 41: End
L0009:  GOTO L0009
; 42:
; 43: On Interrupt 'Timer0 interrupt routine
L0006:
; 44:  'continuously switch between high and low digit displays
; 45: If phase = 0 Then
        BTFSC 0x2C,0
        GOTO L0010
; 46: phase = 1
        BSF 0x2C,0
; 47: Gosub show1
        CALL L0003
; 48: Else
        GOTO L0011
L0010:  MOVLW 0x1F
        ANDWF STATUS,F
; 49: phase = 0
        BCF 0x2C,0
; 50: Gosub show2
        CALL L0004
; 51: Endif
L0011:  MOVLW 0x1F
        ANDWF STATUS,F
; 52: INTCON.T0IF = 0 'enable new TMR0 interrupts
        BCF 0x0B,2
; 53: Resume
        RETURN
; 54:
; 55: getmask:  'get appropriate 7-segment mask for input digit
L0002:
; 56: mask = LookUp(0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f), digit
        MOVF 0x2D,W
        SUBLW 0x09
        BTFSS STATUS,C
        GOTO L0012
        CALL L0013
        MOVWF 0x30
```

```
        GOTO L0012
L0013:
        MOVLW 0x00
        MOVWF PCLATH
        MOVF 0x2D,W
        ADDWF PCL,F
        RETLW 0x3F
        RETLW 0x06
        RETLW 0x5B
        RETLW 0x4F
        RETLW 0x66
        RETLW 0x6D
        RETLW 0x7D
        RETLW 0x07
        RETLW 0x7F
        RETLW 0x6F
L0012:
; 57: Return
        RETURN
; 58:
; 59: show1:  'show high digit on its display
L0003:
; 60: d2enable = False
        BCF 0x07,1
; 61: PORTB = mask1
        MOVF 0x31,W
        MOVWF 0x06
; 62: d1enable = True
        BSF 0x07,0
; 63: Return
        RETURN
; 64:
; 65: show2:  'show low digit on its display
L0004:
; 66: d1enable = False
        BCF 0x07,0
; 67: PORTB = mask2
        MOVF 0x32,W
        MOVWF 0x06
; 68: d2enable = True
        BSF 0x07,1
; 69: Return
        RETURN
; End of program
L0014:GOTO L0014
; Division Routine
D001:  MOVLW 0x10
        MOVWF R3L
        CLRF R2H
        CLRF R2L
D002:  RLF R0H,W
        RLF R2L,F
```

```
        RLF R2H,F
        MOVF R1L,W
        SUBWF R2L,F
        MOVF R1H,W
        BTFSS STATUS,C
        INCFSZ R1H,W
        SUBWF R2H,F
        BTFSC STATUS,C
        GOTO D003
        MOVF R1L,W
        ADDWF R2L,F
        MOVF R1H,W
        BTFSC STATUS,C
        INCFSZ R1H,W
        ADDWF R2H,F
        BCF STATUS,C
D003:  RLF R0L,F
        RLF R0H,F
        DECFSZ R3L,F
        GOTO D002
        MOVF R0L,W
        RETURN
; Waitus Routine - Word Argument
Y001:  MOVLW 0x10
        SUBWF R4L,F
        CLRW
        BTFSS STATUS,C
        ADDLW 0x01
        SUBWF R4H,F
        BTFSS STATUS,C
        RETURN
        GOTO Y002
Y002:  MOVLW 0x0A
        SUBWF R4L,F
        CLRW
        BTFSS STATUS,C
        ADDLW 0x01
        SUBWF R4H,F
        BTFSS STATUS,C
        RETURN
        GOTO Y002
; End of listing
        END
```

3. Послідовність роботи з симулятором при виконанні програми

Виконаємо цю програму в PIC Simulator ID, для чого необхідно:

1. Запустити PIC Simulator IDE;
2. Натиснути Options | Select Microcontroller;
3. Вибрати PIC16F84 і натиснути кнопку Select;
4. В папці «Program Files» Вашого комп'ютера знайти папку «PIC Simulator IDE» з інсталяцією симулятора;
5. Відкрити файл «7segment.asm» і скопіювати його вміст;

6. Натиснути Tools і у вікні, що розкриється, вибрати «Assembler». Відкриється вікно компілятора «Assembler – UNTITLED» (рис. 2);

7. Вставити скопійований раніше файл «7segment.asm» у вікно «Assembler»;

8. Натиснути Tools і у вікні, що розкриється – Assemble. В нижній половині вікна Assembler з'явиться лістинг відкомпільованого файлу (рис. 6);

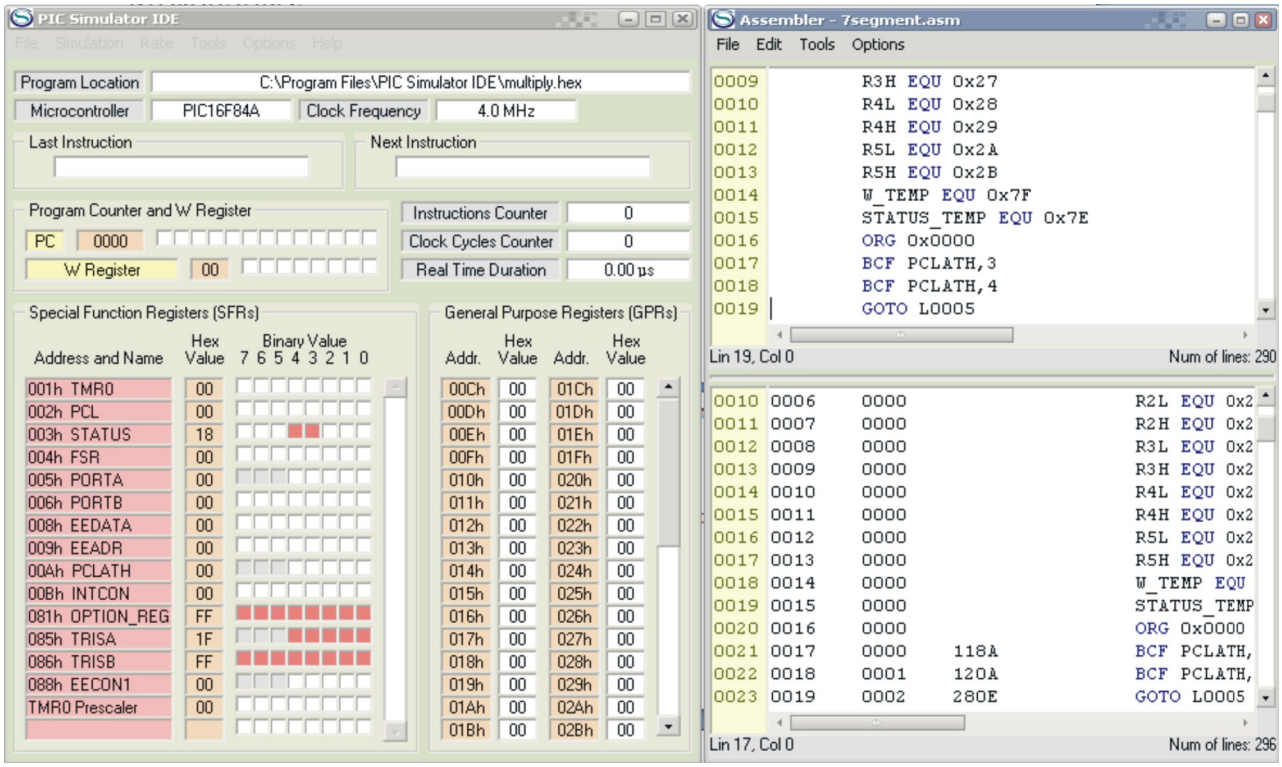

Рис. 6 Вигляд екрану з програмою «Вивід даних на 7-сегментний дисплей»

9. Одночасно, при відсутності помилок, буде створений файл «7segment.hex», для якого можна вибрати ім'я та шлях для запису. Записати його на «Робочий стіл» комп'ютера;

10. Вибрати File | Load Program і завантажити створений файл «7segment.hex»;

11. Натиснути Tools | 7- Segment LED Displays Panel (відкриється вікно з чотирма 7-сегментними дисплеями);

12. Натиснути кнопку Setup під дисплеєм номер "2" (нумерація починається справа);

13. Натиснути помаранчеве поле під дисплеєм, щоб включити його («Display Enable») або виключити цей дисплей;

14. Натиснути на жовте поле справа від помаранчевого (з написом «Disable»). З'явиться на ньому напис "Always Anabled». Натиснути на нього ще раз – справа відкриється вікно «Select Pin»;

15. У вікні «Select Pin» почергово натиснути поле «PORTB» і далі «0», після чого натиснути на поле «Select», яке розташоване внизу вікна. Таким чином, вибрано порт В та його вивід RB0 (рис. 7);

16. Натиснути кнопку Setup під дисплеєм номер "1" (нумерація починається справа);

17. Натиснути помаранчеве поле під дисплеєм, щоб включити його («Display Enable») або виключити цей дисплей;

18. Натиснути на жовте поле справа від помаранчевого (з написом «Disable»). З'явиться на ньому напис "Always Anabled». Натиснути на нього ще раз – справа відкриється вікно «Select Pin»;

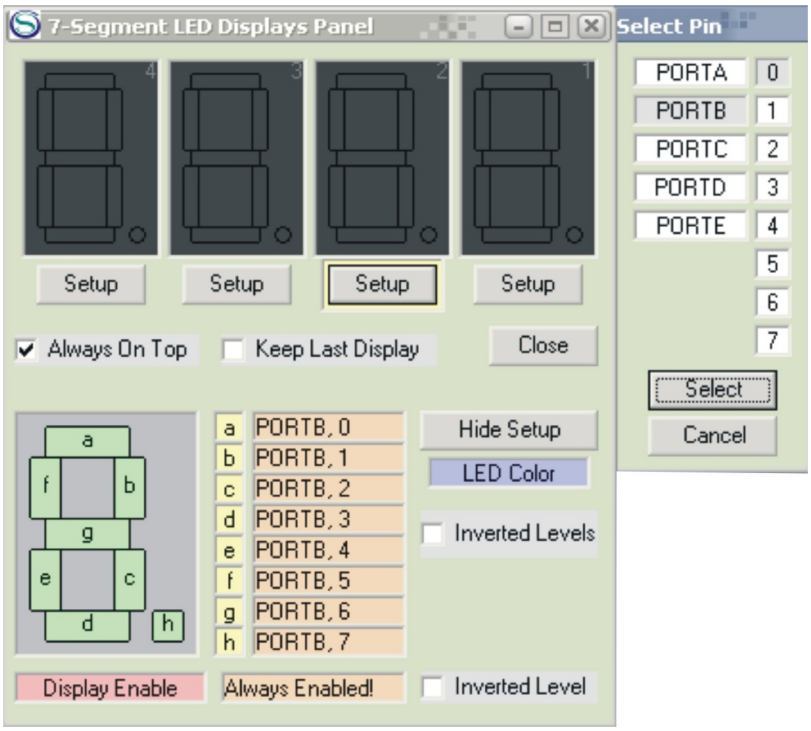

Рис. 7 Налаштування другого справа розряду індикатора для виводу даних

19. У вікні «Select Pin» почергово натиснути поле «PORTB» і далі «1», після чого натиснути на поле «Select», яке розташоване внизу вікна. Таким чином, вибрано порт B та його вивід RB1;

20. Натиснути, при потребі, кнопку Hide Setup, щоб зберегти трохи екранного простору;

21. Вибрати Rate | Ultimate (No Refresh);

22. Натиснути Simulation | Start (почнеться виконання програми). Програма відобразить числа від 0 до 99 на двох 7-сегментних дисплеях, використовуючи процедуру мультиплексування переривання TMR0.

Зберігаючи дані на екрані, необхідно вибрати опцію Keep Last Display.

Вигляд екрану з виконуваною програмою показаний на рис. 8.

23. Щоб зупинити виконання програми, потрібно натиснути Simulation | Stop.

Для того, щоб мати змогу контролювати вміст регістрів після виконання стимулятором кожної команди, перейти на виконання програми в крокову режимі роботи. Для цього:

1. В основному вікні симулятора натиснути Rate | Step By Step, а далі вибрати опцію Simulation і натиснути Start. Симулятор готовий до виконання програми в крокову режимі;

2. Для виконання наступної команди програми потрібно натиснути на закладку STEP, яка з'явиться справа від закладки HELP вгорі основного вікна симулятора після вибору крокового режиму його роботи.

Вміст регістрів контролера, які використовуються при виконанні команд програми, знайти в області регістрів Adress and Name, яка розташована в лівій нижній частині основного вікна симулятора (виділені рожевим кольором). Всі регістри восьмирозрядні.

В процесі виконання програми по зміні кольору комірок видно, вміст яких регістрів змінюється. Забарвлення комірки відповідного розряду регістру помаранчевим кольором означає наявність "1", білим - "0".

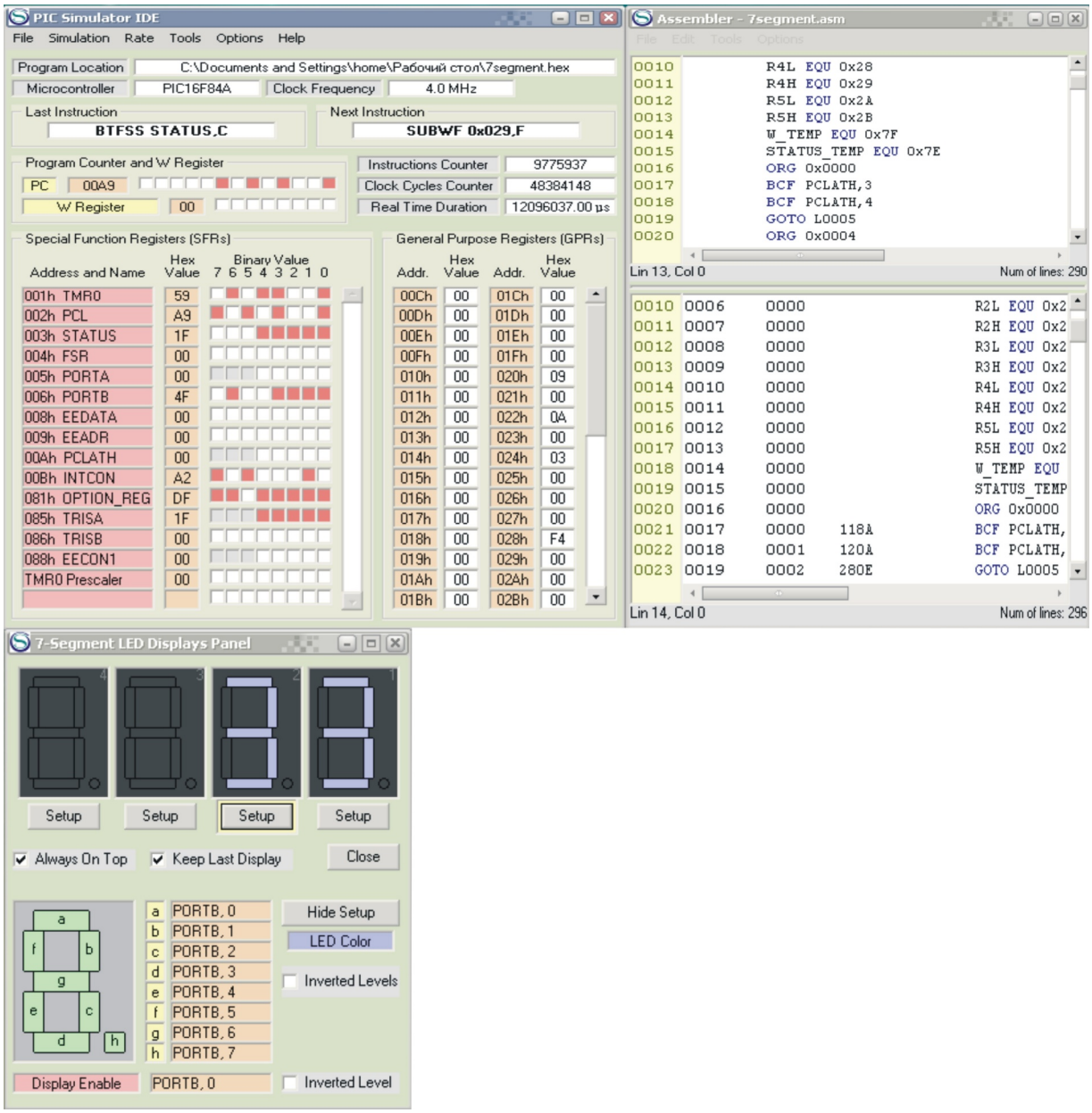

Рис. 8 Вивід даних на 7- сегментний дисплей

Завдання

1. Вміст тих регістрів, значення яких змінюється в процесі виконання команд програми, записати в шістнадцятковому коді в табл.1.

Таблиця 1

Регістр	PC	W	TMR0	STATUS	PCL	TRISA	TRISB	PCLATH	EEADR	EEDATA	FSR
Команда 1											
Команда 2											
……..											
Команда n											

2. З вікна з програмою (рис. 7) вибрати десять команд і за таблицею команд асемблера для PIC-контролера (табл. 1) записати коментар щодо призначення цих команд (див. Приклад 2, де наведено такий запис для однієї команди).

Приклад 2

Код команди	**Команда**	**Виконувана операція (коментар)**
118A	BCF PCLATH, 3	; скинути в “0” 3-ій біт регістра PCLATH

і т.д.

4. Контрольні запитання

1. Будова PIC-контролерів серії PIC16X8X.
2. Призначення регістра W мікроконтролера.
3. Формат та призначення регістрів спеціальних функцій.
4. Як програмуються лінії портів на ввід та на вивід?
5. Формат регістра STATUS.
6. Призначення та позначення основних елементів програмної моделі мікроконтролера.

5. Література

1. Данилин А. Программа-симулятор PIC Simulator IDE / Данилин А. // Современная электроника. 2006.- №4. -С. 68-76.

2. Тавернье К. PIC-микроконтроллеры. Практика применения. М.: ДМК, 2002.

3. Предко М. Создайте робота своими руками на PIC- контроллере./ Майкл Предко; Пер. с английского Земского Ю.В. – М.: ДМК Пресс, 2006. – 408 с.: ил.

4. Кениг А. и М. Полное руководство по PIC-микроконтроллерам.: Пер. с нем.-К.: МК- Пресс”, 2007.-256 с., ил.

Таблиця 2. Команди контролера PIC16X8X

Позначення	Функція	Цикли	Код команди	Біти стану	Прим.
ADDLW	Додавання константи і W	1	11111x	C, DC, Z	
ADDWF	Додавання W с f	1	00 0111 dfff ffff	C, DC, Z	1, 2
ANDLW	Логічне І константи і W	1	11 1001 kkkk kkkk	Z	
ANDWF	Логічне І W і f	1	00 0101 dfff ffff	Z	1, 2
BCF	Скидання біту в регістрі f	1	01 00bb bfff ffff		1, 2
BSF	Встановлення біту в регістрі f	1	01 01bb bfff ffff		1, 2
BTFSC	Пропустити команду, якщо біт у f дорівнює нулю	1 (2)	01 10bb bfff ffff		3
BTFSS	Пропустити команду, якщо біт у f дорівнює одиниці	1 (2)	01 11bb bfff ffff		3
CALL	Виклик підпрограми	2	10 0kkk kkkk kkkk		
CLRF	Скидання регістру f	1	00 0001 1fff ffff	Z	2
CLRW	Скидання регістра W	1	00 0001 0xxx xxxx	Z	
CLRWDT	Скидання сторожового таймера WDT	1	00 0000 0110 0100		
COMF	Інверсія регістру f	1	00 1001 dfff ffff	Z	1, 2
DECF	Декремент регістру f	1	00 0011 dfff ffff	Z	1, 2
DECFSZ	Декремент f, пропустити команду, якщо 0	1 (2)	00 1011 dfff ffff		1, 2, 3
GOTO	Перехід за адресою	2	10 1kkk kkkk kkkk		
INCF	Інкремент регістру f	1	00 1010 dfff ffff	Z	1, 2
INCFSZ	Інкремент f, пропустити команду, якщо 0	1 (2)	00 1111 dfff ffff		1, 2, 3
IORLW	Логічне АБО константи і W	1	11 1000 kkkk kkkk	Z	
IORWF	Логічне АБО W і f	1	00 0100 dfff ffff	Z	1, 2
MOVF	Пересилання регістру f	1	00 1000 dfff ffff	Z	1, 2
MOVLW	Пересилання константи в W	1	11 00xx kkkk kkkk		
MOVWF	Пересилання W у f	1	00 0000 1fff ffff		
NOP	Пуста команда	1	00 0000		

OPTION	Завантаження регістру OPTION	1	00 0000 0110 0010		
RETFIE	Повернення з переривання	2	00 0000 0000 1001		
RETLW	Повернення з підпрограми з завантаженням константи в W	2	11 01xx kkkk kkkk		
RETURN	Повернення з підпрограми	2	00 0000 0000 1000		
RLF	Зсув f вліво через перенесення	1	00 1101 dfff ffff	C	1, 2
RRF	Зсув f вправо через перенесення	1	00 1100 dfff ffff	C	1, 2
SLEEP	Перехід у режим SLEEP	1	00 0000 0110 0011		
SUBLW	Вирахування W з константи	1	11 110x kkkk kkkk	C, DC, Z	
SUBWF	Вирахування W з f	1	00 0010 dfff ffff	C, DC, Z	1, 2
SWAPF	Обмін місцями тетрад в f	1	00 1110 dfff ffff		1, 2
TRIS	Завантаження регістру TRIS	1	00 0000 0110 0fff		
XORLW	Виключаюче АБО константи і W	1	11 1010 kkkk kkkk	Z	
XORWF	Виключаюче АБО W і f	1	00 0110 dfff ffff		1, 2

*Якщо в результаті виконання команди змінюється лічильник команд, або виконується перехід по перевірці умови, то команда виконується за два цикли. Другийцикл виконується як NOP.