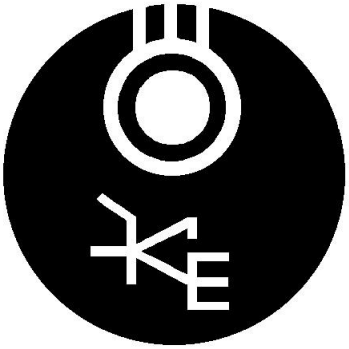
Міністерство освіти і науки України
Національний гірничий університет

П.І.Б. _____
Група _____
Варіант _____
Відмітка про залік:
_____



МЕТОДИЧНІ ВКАЗІВКИ
ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ МПП-3
«БІТОВИЙ ПРОЦЕСОР МІКРОКОНТРОЛЕРА 8051АН СІМЕЙСТВА MCS-51. ПРОГРАМУВАННЯ В ІНТЕГРОВАНОМУ ІНСТРУМЕНТАЛЬНОМУ СЕРЕДОВИЩІ IDE COMPASS/51/251»,

індивідуальних завдань та самостійної роботи
з професійно-орієнтованої дисципліни «Мікропроцесорні пристрої»
для студентів спеціальності 7.092203 «Електромеханічні системи автоматизації та електропривод» напряму «Електромеханіка»

Дніпропетровськ
2006

Міністерство освіти і науки України
Національний гірничий університет

Методичні вказівки
до виконання лабораторної роботи МПП-3 «Бітовий процесор мікроконтролера 8051АН сімейства MCS-51. Програмування в інтегрованому інструментальному середовищі IDE COMPASS/51/251», індивідуальних завдань та самостійної роботи з професійно-орієнтованої дисципліни «Мікропроцесорні пристрої» для студентів спеціальності 7.092203 «Електромеханічні системи автоматизації та електропривод» напряму «Електромеханіка»

Дніпропетровськ, НГУ
2006

Методичні вказівки до виконання лабораторної роботи МПП-3 «Бітовий процесор мікроконтролера 8051АН сімейства MCS-51. Програмування в інтегрованому інструментальному середовищі IDE COMPASS/51/251», індивідуальних завдань та самостійної роботи з професійно-орієнтованої дисципліни «Мікропроцесорні пристрої» для студентів спеціальності 7.092203 «Електромеханічні системи автоматизації та електропривод» напряму «Електромеханіка» / Упоряд.н.: В.І. Кириченко, О.А. Яланський, В.Г. Алпаєв. – Дніпропетровськ: НГУ, 2006. – 51 с.

Упорядники: Кириченко Віталій Іванович, д-р техн. наук;
Яланський Олексій Анатолійович, канд. техн. наук;
Алпаєв Володимир Георгієвич, магістр

Відповідальний за випуск завідувач кафедри електропривода
О.С. Бешта, д-р техн. наук, професор

ЗМІСТ

ВСТУП

Мета самостійної та лабораторної робіт – подальше поглиблення знань з дисципліни «Мікропроцесорні засоби». На даному етапі пропонується вивчення системи команд бітового процесора мікроконтролера 8051 АН сімейства MCS-51 та особливостей його програмування, зокрема, для реалізації обрахунків перемикальних функцій з врахуванням наведених правил алгебри логіки. Приведені приклади програмного керування бітами окремих вузлів (регістрів, елементів пам’яті тощо) вибраного мікроконтролера. Для розробки та налагодження програмного забезпечення конкретного призначення в методичних вказівках подано матеріал щодо особливостей та послідовності виконання цих етапів. Для досягнення мети запропоновано використання інтегрованого інструментального середовища IDE COMPASS/51/251.

1. ЗМІСТ САМОСТІЙНОЇ ТА ЛАБОРАТОРНОЇ РОБІТ

1.1. Самостійна робота

Самостійна робота полягає у вивченні теоретичного матеріалу, який подано у даних методичних вказівках. Слід ознайомитися із елементами алгебри логіки, вивчити систему команд бітового процесора мікроконтролера 8051АН сімейства MCS-51 та особливостей програмування обрахунків перемикальних функцій, ознайомитися із можливостями бітового процесора щодо формування окремих бітів програмно доступних вузлів мікроконтролера. Додатково слід ознайомитися и принципами технології розробки програмного забезпечення із використанням програмних засобів автоматизації, зокрема інтегрованого інструментального середовища IDE COMPASS/51/251.Засвоївши матеріал, необхідно відповісти на запитання самоперевірки, які наведені у кінці відповідного розділу.

1.2. Лабораторна робота

Виконується у комп’ютерних класах кафедри електропривода. До її виконання допускаються студенти, які ознайомилися із теоретичним матеріалом у обсязі, зазначеному в п.1.1 та підготували попередній звіт.

Назва роботи

Бітовий процесор мікроконтролерів 8051АН сімейства MCS-51. Програмування в інтегрованому інструментальному середовищі IDE COMPASS/51/251.

Мета роботи

Кінцевою метою роботи є ознайомлення з особливостями використання команд бітового процесора мікроконтролерів 8051АН сімейства MCS-51 та отримання навичок практичного програмування за допомогою інтегрованого інструментального середовища IDE COMPASS/51/251.

Програма роботи

- Ознайомлення з елементами алгебри логіки.
- Ознайомлення з системою команд та особливостями використання бітового процесора, побудовою прикладних програм.
- Ознайомлення з інтегрованим інструментальним середовищем IDE COMPASS/51/251 та прийомами роботи з ним при розробці та налагодженні програмного забезпечення бітового процесора.
- Отримання відповідей на запитання самоперевірки рівня засвоєння теоретичного матеріалу.
- Виконання індивідуального завдання та демонстрація викладачеві результату використання розробленої програми як допуск до захисту лабораторної роботи шляхом тестової перевірки.
- Укладання заключного звіту та захист роботи шляхом тестової перевірки.

Вказівки щодо укладання звіту

Попередній звіт повинен містити:

- Назву, мету та програму роботи.
- Відповіді на запитання самоперевірки.
- Відповіді на запитання індивідуального варіанту самостійної роботи.
- Чернетку з алгоритмом та програмою індивідуального завдання вибраного рівня складності.

Підсумковий звіт, окрім перших трьох пунктів попереднього звіту, повинен містити алгоритм та налагоджену програму індивідуального завдання вибраного рівня складності, перевірені та підписані викладачем.

Увага! Допуском до виконання лабораторної роботи є підготовлений попередній звіт згідно поставлених до нього вимог.

2. АЛГЕБРА ЛОГІКИ І БІТОВИЙ ПРОЦЕСОР

2.1. Загальні відомості з алгебри логіки

У середині минулого століття ірландський математик Д. Буль розробив математичний апарат для вирішення завдань формальної логіки. У подальшому запропонована ним алгебра логіки на вшанування його імені отримала назву булевої алгебри.

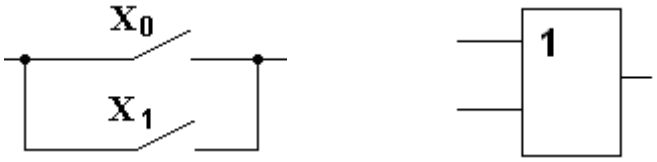
Булева алгебра — це математична система, що оперує двома поняттями: подія істинна і подія неістинна. Природно асоціювати ці поняття з цифрами, двійкової системи числення. Два елементи булевої алгебри, а саме подія істинна і подія неістинна називають її константами. Розумітимемо під ними значення відповідно 1 і 0.

Серед основних операцій булевої алгебри операції логічного додавання, множення і заперечення.

Логічне додавання. Цю операцію називають «АБО» (*диз'юнкція)*. Постулати логічного логічного додавання двох змінних наведені у вигляді даних таблиці істинності

X_1	X_0	$X_1 \vee X_0$ (X_1+X_0)
0	0	0
0	1	1
1	0	1
1	1	1

та у вигляді схемотехнічних аналогів

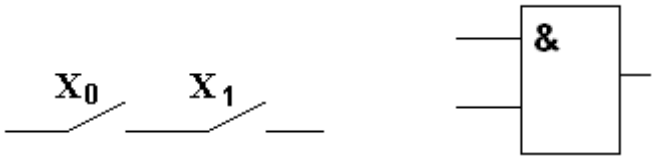


Слід зазначити, що операція справедлива для довільного числа змінних.

Логічне множення. Цю операцію називають операцією «І» або *кон'юнкцією.* Постулати логічного множення двох змінних наведені у вигляді даних таблиці істинності

X_1	X_0	$X_1 \wedge X_0$ ($X_1 \cdot X_0$)
0	0	0
0	1	0
1	0	0
1	1	1

та схемотехнічних аналогів



Ця операція також справедлива для довільного числа змінних.

Заперечення. Операцію заперечення називають «НІ», *інверсією* або *доповненням.* Для її позначення використовують риску над відповідним виразом. Операція «НІ» визначається наступними постулатами, а саме:

якщо X = 1, то $\overline{X} = 0$, а якщо X = 0, то $\overline{X} = 1$.

Важливим практичним наслідком є той факт, що при запису логічних виразів і побудові логічних схем можливо обійтися лише двома типами операцій, наприклад, операціями «І» і «НІ» або «АБО» і «НІ».

2.2. Основні правила алгебри логіки

В алгебрі логіки розрізняють основні і допоміжні закони. Є чотири основні закони:

1. **Комутативний** (закон перестановки)

$$X_1 \vee X_2 = X_2 \vee X_1 \text{ – для диз'юнкції;}$$
$$X_1 \wedge X_2 = X_2 \wedge X_1 \text{ – для кон'юнкції.}$$

2. **Асоціативний** (закон сполучення)

$$(X_1 \vee X_2) \vee X_3 = X_1 \vee (X_2 \vee X_3) \text{ – для диз'юнкції;}$$
$$(X_1 \wedge X_2) \wedge X_3 = X_1 \wedge (X_2 \wedge X_3) \text{ – для кон'юнкції.}$$

3. **Дистрибутивний** (закон розподілу)

$$(X_1 \vee X_2) \wedge X_3 = X_1 \wedge X_3 \vee X_2 \wedge X_3 \text{ – для диз'юнкції;}$$
$$X_1 \wedge X_2 \vee X_3 = (X_1 \vee X_3) \wedge (X_2 \vee X_3) \text{ – для кон'юнкції.}$$

Перевірка цього закону для кон'юнкції для X1 = 1; X2 = 1; X3 = 0:

ліва частина $1 \wedge 1 \vee 0 = 1;$
права частина $(1 \vee 1) \wedge (1 \vee 0) = 1.$

4. **Інверсії** (правило Моргана)

$$\overline{X_1 \vee X_2} = \overline{X_1} \wedge \overline{X_2} \text{ – для диз'юнкції;}$$
$$\overline{X_1 \wedge X_2} = \overline{X_1} \vee \overline{X_2} \text{ – для кон'юнкції.}$$

2.3. Допоміжні закони алгебри логіки

1. **Подвійної інверсії** (подвійного заперечення)

$$\overline{\overline{X}} = X.$$

2. **Повторення** (індемпотентності)

$$X \vee X \vee X \ldots = X; \; X \wedge X \wedge X \ldots = X.$$

3. **Суперечності**

$$X \wedge \overline{X} = 0.$$

4. **Виключення третього**

$$X \vee \overline{X} = 1.$$

5. **Склеювання**

$$X_1 \wedge \overline{X}_2 \vee X_1 \wedge X_2 = X_1;$$
$$X_1 \vee \overline{X}_2 \wedge X_1 \vee X_2 = X_1$$

6. **Поглинання**

$$X_1 \vee X_1 \wedge X_2 = X_1;$$
$$X_1 \wedge (X_1 \vee X_2) = X_1.$$

7. **Незмінності**

$$X \vee 0 = X;$$
$$X \wedge 1 = X.$$

8. **Універсальної і нульової множини**

$$X \vee 1 = 1;$$
$$X \wedge 0 = 0.$$

За допомогою цих законів можна виконувати різні перетворення булевих функцій дотримуючись такого порядку: першими виконуються операції в дужках; за їх відсутності першими виконуються операції заперечення (інверсії), потім кон'юнкції і лише потім диз'юнкції.

2.4. Мінімізація булевих функцій

Мінімізація булевих функцій виконується для спрощення булевих виразів, що у підсумку скорочує довжину програм та зменшує термін їх виконання (при апаратній реалізації виразів це зменшує кількість елементів схеми). Розрізняють аналітичні і табличні методи мінімізації.

Табличні методи базуються на використанні карт Карно і Вейча, а аналітичні – на використанні законів перетворення булевих виразів.

Приклад 1.

$$f(X_1, X_2, X_3) = X_1 \wedge \overline{X}_2 \wedge X_3 \vee \overline{X}_1 \wedge \overline{X}_2 \wedge X_3 \vee X_1 \wedge \overline{X}_2 \wedge \overline{X}_3 \vee X_1 \wedge X_2 \wedge \overline{X}_3.$$

Після об'єднання попарно першого з другим та третього з четвертим добутком отримуємо вираз

$$f(X_1, X_2, X_3) = (X_1 \wedge \overline{X}_2 \wedge X_3 \vee \overline{X}_1 \wedge \overline{X}_2 \wedge X_3) \vee (X_1 \wedge \overline{X}_2 \wedge \overline{X}_3 \vee X_1 \wedge X_2 \wedge \overline{X}_3),$$

що після винесення за дужки і спрощення дає значно простіший булевий вираз наступного вигляду

$$f(X_1, X_2, X_3) = \overline{X}_2 \wedge X_3(\overline{X}_1 \wedge X_1) \vee X_1 \wedge \overline{X}_3 \wedge (X_2 \wedge \overline{X}_2) = \overline{X}_2 \wedge X_3 \vee X_1 \wedge \overline{X}_3.$$

Приклад 2

У таблиці 2.1 наведені дані щодо переходів синхронного R-S тригера із трьома входами C, R, S і одним (прямим) виходом до (tn, Qn) та після (tn+1, Qn+1) момента перемикання бістабільного елемента на рис. 2.1.

Таблиця 2.1

C	t_n			t_{n+1}		
	R	S	Q_n	Q_{n+1}	f_1	f_2
0	0	0	0	0	1	*
0	0	0	1	1	*	1
0	0	1	0	0	1	*
0	0	1	1	1	*	1
0	1	0	0	0	1	*
0	1	0	1	1	*	1
0	1	1	0	0	1	*
0	1	1	1	1	*	1
1	0	0	0	0	1	*
1	0	0	1	1	*	1
1	0	1	0	1	0	1
1	0	1	1	1	*	1
1	1	0	0	0	1	*
1	1	0	1	0	1	0
1	1	1	0	*	0	0
1	1	1	1	*	0	0

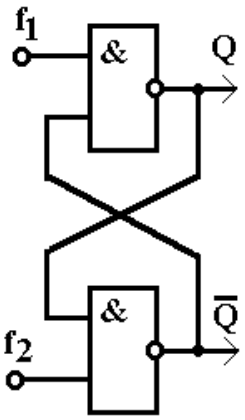


Рис. 2.1. Схема бістабільної ланки

Тут f_1, f_2 – функції збудження бістабільної кон'юнктивної ланки, яка є основою для побудування тригерів різного типу. При використанні цього методу спочатку на основі даних табл. 1 послідовно заповнюють клітинки діаграм Вейча для функцій f1 (ліворуч) та f2 (праворуч) відповідно, причому значення «*» означає, що функція може приймати значення 0 або 1 (рис. 2.2).

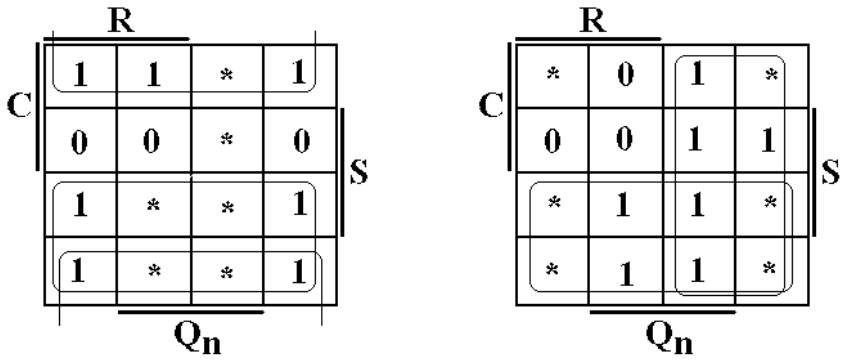


Рис. 2.2. Розрахункова схема бістабільної ланки

У підсумку після «склеювань» ділянок діаграм Вейча за відповідними правилами [1] перемикальні функції збудження f1 та f2 приймають остаточний вигляд:

$$f_1 = \underline{C \wedge S};$$
$$f_2 = \underline{C \wedge R},$$

а схема синхронного R-S тригера набрала вигляду рис. 2.3.

Отже, завдяки використанню діаграм Вейча, завдання отримати мінімальну диз'юнктивну нормальну форму (МДНФ) – найпростішого вигляду логічного виразу – значно спрощується.

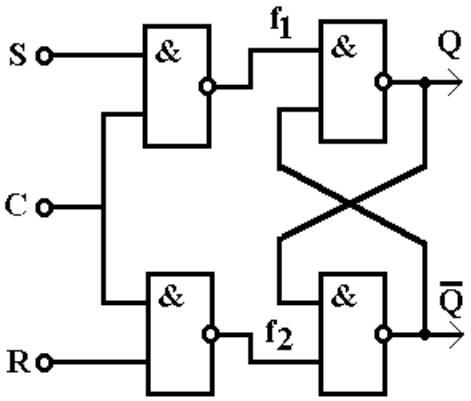


Рис. 2.3. Схема синхронного R-S тригера

2.5. Бітовий процесор і структура системи команд

Для керування об'єктами часто використовують алгоритми, які містять операції над булевими змінними (істина/неістина). Важливою особливістю АЛУ мікроконтролера є його здатність оперувати із окремими бітами. Він має 17 команд для виконання таких операцій (табл. 2.2). Програмно доступні біти можуть бути встановлені в 1, скинуті в 0, проінвертовані, передані, перевірені і використані в логічних операціях. Таку здатність мікроконтролера часто називають його бітовим процесором.

Таблиця 2.2

Мнемоніка команди	Опис команди	Код команди	Кількість байтів	Кількість циклів
SETB C	Встановлення ознаки перенесення	0D3h	1	1
SETB bit	Встановлення біту	0D2h	2	1
CLR C	Скидання ознаки перенесення	0C3h	1	1
CLR bit	Скидання біту	0C2h	2	1
CPL C	Інверсія ознаки перенесення	0B3h	1	1
CPL bit	Інверсія біту	0B2h	2	1
MOV C, bit	Пересилання біту в ознаку перенесення	0A2h	2	1
MOV bit, C	Пересилання ознаки перенесення в біт	92h	2	2
ANL C, bit	Логічне “І” біту та ознаки перенесення	82h	2	2
ANL C, /bit	Логічне “І” інверсії біту та ознаки перенесення	0B0h	2	2
ORL C, bit	Логічне “АБО” біту і ознаки перенесення	72h	2	2
ORL C, /bit	Логічне “АБО” інверсії біту і ознаки перенесення	0A0h	2	2
JC rel8	Перехід, якщо ознака перенесення встановлена	40h	2	2
JNC rel8	Перехід, якщо ознака перенесення не встановлена	50h	2	2
JB bit, rel8	Перехід, якщо bit встановлений	20h	3	2
JNB bit, rel8	Перехід, якщо bit встановлений	30h	3	2
JBC bit, rel8	Перехід, якщо bit встановлений і скидання цього біту після виконання команди	10h	3	2

В табл. 2.2 прийнято наступні позначення:

C — ознака (*прапор) перенесення;* bit — *один із 128 програмно-доступних бітів ОЗП, будь-який I/O вивід, біти керування або програмно-доступні біти байта стану;* /bit — *інверсія одного із 128 програмно-доступних бітів ОЗП, будь-якого I/O виводу, бітів керування або бітів байта стану;* rel8 — *байт відносного зсуву (умовний перехід здійснюється в діапазоні від -128 до +127 байтів відносно адреси першого байту наступної команди).*

Зазначені команди залежно від функції можуть бути одно-, дво- та трибайтними. За допомогою цих команд можна звертатися безпосередньо до 128 бітів внутрішнього ОЗП (оперативного запам'ятовуючого пристрою, рис. 2.4) та 83 бітів одинадцяти восьмирозрядних регістрів мікроконтролера (рис. 2.5).

Адреси байтів	Біти							
	D7	D6	D5	D4	D3	D2	D1	D0
2FH	7F	7E	7D	7C	7B	7A	79	78
2EH	77	76	75	74	73	72	71	70
2DH	6F	6E	6D	6C	6B	6A	69	68
2CH	67	66	65	64	63	62	61	60
2BH	5F	5E	5D	5C	5B	5A	59	58
2AH	57	56	55	54	53	52	51	50
29H	4F	4E	4D	4C	4B	4A	49	48
28H	47	46	45	44	43	42	41	40
27H	3F	3E	3D	3C	3B	3A	39	38
26H	37	36	35	34	33	32	31	30
25H	2F	2E	2D	2C	2B	2A	29	28
24H	27	26	25	24	23	22	21	20
23H	IF	1E	ID	1C	1B	1A	19	18
22H	17	16	15	14	13	12	11	10
21H	0F	0E	0D	0C	0B	0A	09	08
20H	07	06	05	04	03	02	01	00

Рис. 2.4. Бітова адресація ОЗП

Пряма адреса байтів	Біти								Ідентифікатори регістрів
	D7	D6	D5	D4	D3	D2	D1	D0	
0F0H	F7	F6	F5	F4	F3	F2	Fl	F0	B
0E0H	E7	E6	E5	E4	E3	E2	El	E0	ACC
	C	AC	F0	RS1	RS0	0V	1	P	
0D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
				PS	PT1	PX1	PT0	PX0	
0B8H	-	-	-	BC	BB	BA	B9	B8	IP
0B0H	B7	B6	B5	B4	B3	B2	Bl	B0	P3
	EA			ES	ET1	EX1	ET0	EX0	
0A8H	AF	-	-	AC	AB	AA	A9	A8	IE
0A0H	A7	A6	A5	A4	A3	A2	Al	A0	P2
	SM0	SM1	SM2	REN	TB8	RB8	T1	R1	
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	P1
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
80H	87	86	85	84	83	82	81	80	P0

Рис. 2.5. Адреси бітів регістрів спеціальних функцій

У межах адрес 20…2Fh ОЗП кожен із бітів зазначених байтів має свою адресу у межах 00…7Fh (00…12810). Адреси 80…0FFh відповідають регістрам спеціальних функцій. Адреси їх бітів формуються із 5 старших бітів адреси регі-

стра та 3 молодших бітів, які визначають адресу біта у межах регістру. Разом з тим не всі біти регістрів спеціальних функцій мають власну адресу. У 6 з них розряди мають власні символічні імена (рис. 2.5) і до них можна звертатися з використанням відповідної мнемоніки (C, AC, F0 тощо) або за допомогою визначення позиції біту (PSW.7, ACC.2, B.3, 20h.4 тощо). Регістр ACC (акумулятор) і регістр B відносяться до байтової арифметики, однак всі їх біти можуть бути використані і у операціях з бітами, як і всі 32 виводи портів P0…P3. *Використання бітів, які на рис. 2.5 замість адреси мають символ «-» або значення «1», у бітових операціях не допускається.* Для опрацювання бітів регістрів ОЗП, які не мають власних адрес, можливе використання логічних операцій над байтами.

Серед команд, поданих у табл. 2.2 та на рис. 2.4 і рис. 2.5, можна виділити наступні групи.

Управління станом

Двобайтні команди SETB, CLR і CPL встановлюють, скидають в нульовий стан або логічно доповнюють до 2 (для бітового операнда це еквівалентно інверсії) біти, що адресуються (або прапори) за один машинний цикл. Команди SETB і CLR здійснюють завантаження біта константою 1 або 0 відповідно. Однобайтні версії цих команд виконують ті ж операції з бітом перенесення C. На мові Асемблера адреси бітів у командах бітового процесора можуть визначатися, наприклад, так:

```
USR_FLG BIT PSW.5     ; Опис символу користувача
;           …     …
CLR 0D5h          ; Використання абсолютної адресації
CLR PSW.5         ; Використання прямої адреси
CLR F0            ; Використання власного імені
CLR USR_FLG       ; Використання символу користувача
```

Пересилка даних

Двобайтові команди MOV за один машинний цикл можуть передавати будь-який біт, що адресується, в ознаку перенесення C або копіювати її в будь-який біт за два цикли. Біт може бути переміщений з одного елемента в інший за допомогою комбінації цих двох команд. Ознака (прапор) перенесення використовується як однорозрядний регістр-акумулятор для логічних операцій. Значення ознаки перенесення можна зберегти командами PUSH і POP.

Логічні операції

Команди ANL і ORL виконують операції логічних "І" та "АБО" відповідно між бітом перенесення C і будь-яким іншим бітом і розміщують отриманий результат в біт перенесення. Наявність символу "/" перед операндом джерела вказує на використання операції логічного доповнення до двох (інверсії) біту, який адресується, але це не впливає на операнд як джерело біту.

Команди перевірки бітів

Команди умовних переходів JC rel8 (перехід за ознаки C=1) і JNC rel8 (перехід за ознаки C=0) перевіряють стан ознаки (прапора) перенесення C і якщо вона встановлена в 1 або 0 відповідно, то вони передають керування команді, яка розташована за адресою rel8.

Команди JB bit, rel8 (перехід за умови bit = 1) і JNB bit, rel8 (перехід за умови bit = 0) перевіряють значення bit і виконують умовний перехід на адресу rel8. Команда JBC bit, rel8 поєднує операції умовного переходу за станом bit = 1 та наступне скидання цього біту в 0. Всі команди умовного переходу виконуються за два машинні цикли. Останній байт команди є кодом зміщення із знаком в інтервалі від -128 до +127.

Взаємодія з іншими командами

На ознаку перенесення C впливають команди додавання та віднімання, множення та ділення, десяткової корекції та зсуву з урахуванням ознаки перенесення, команди порівняння.

2.6. Використання команд бітового процесора

Використання команд бітового процесора корисне при розрахунках перемикальних функцій, які можуть приймати логічні значення 0 або 1. На рис. 2.6 наведене апаратне (на ТТЛ-елементах) вирішення завдання визначення перемикальної функції Q від шести змінних (U, V, W, X, Y, Z) вигляду

$$Q = U * (V + W) + (X * \underline{Y}) + \underline{Z}.$$

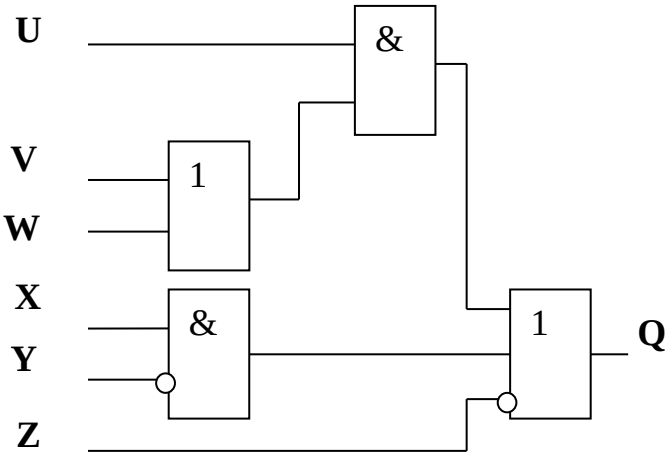


Рис. 2.6. Апаратне визначення перемикальної функції

При вирішенні цього завдання програмними засобами із застосуванням команд бітового процесора припускаємо, що U, V, W, X, Y і Z є вхідними сигналами на виводах порту P2.0… P2.5. Приймаємо також, що після програмного визначення перемикальна функція Q повинна бути виданою на вивід P3.6 порту P3.

При вирішенні поставленого завдання попередньо кожній вхідній змінній надане певне символічне ім'я. По завершенню виконання програми одержаний результат у вигляді отриманого значення перемикальної функції фіксується в ознаці (прапорі) перенесення, який і передається на вихід. Програма QFUN, яка визначає перемикальну функцію Q від 6 змінних, з використанням команд бітового процесора набирає вигляду:

```
      U     BIT   28H   ; Для зручності у відповідність значенню
      V     BIT   29H   ; вхідних сигналів встановлюємо
      W     BIT   2AH   ; біти елемента пам’яті з адресою 25h
      X     BIT   2BH   ;
      Y     BIT   2CH   ;
      Z     BIT   2DH   ;
      Q     BIT   P3.6  ;
START:
      MOV   25H, P2     ; Зчитування значень вхідних сигналів
      MOV   C, V        ; Занесення в біт C рівня сигналу V
      ORL   C, W        ; Логічна сума V+W в ознаці C
      ANL   C, U        ; Логічний добуток U*(V+W) в C
      MOV   F0, C       ; Збереження отриманого добутку
                        ; в біті B.0 регістра B з адресою F0h
      MOV   C, X        ; Занесення в біт C рівня сигналу X
      ANL   C, /Y       ; Логічний добуток X*У
      ORL   C, F0       ; Збереження в C суми добутків
      ORL   C, /Z       ; Збереження в C суми трьох доданків
      MOV   Q, C        ; Виведення перемикальної функції
      JMP   START       ; Перехід на початок програми
```

Отже за допомогою використання команд бітового процесора можливе програмне моделювання будь-якої комбінаційної схеми з N-числом входів. Обмеження при моделюванні комбінаційної схеми з великою кількістю входів і виходів визначається кількістю наявних портів.

3. ТЕХНОЛОГІЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Загальні положення

На сьогодні для розробки і відлагодження об’єктно-орієнтованого програмного забезпечення найчастіше використовують інтегровані інструментальні середовища (Integrated Development Environment, IDE) – програмні оболонки, які працюють під керуванням оператора і операційної системи на персональній ЕОМ загального призначення. Склад типового інтегрального інструментального середовища (ІІС) визначається переліком необхідних операцій для створення програмного продукту. Типовими є наступні операції:

– введення тексту початкової програми на мові програмування високого або низького рівня;

– трансляція початкової програми з метою отримання її у вигляді кодів (об’єктних модулів), які записуються у пам’ять програм;

– об'єднання отриманих в результаті трансляції об'єктних модулів в єдиний завантажувальний модуль, призначений для запису у пам'ять програм;

– налагоджування завантажувального модуля за допомогою програмних або апаратних засобів.

Схема підготовки програм за допомогою інтегрованого інструментального середовища (COMPASS/51/251) приведена на рис. 3.1, а його структура та склад – на рис. 3.2.

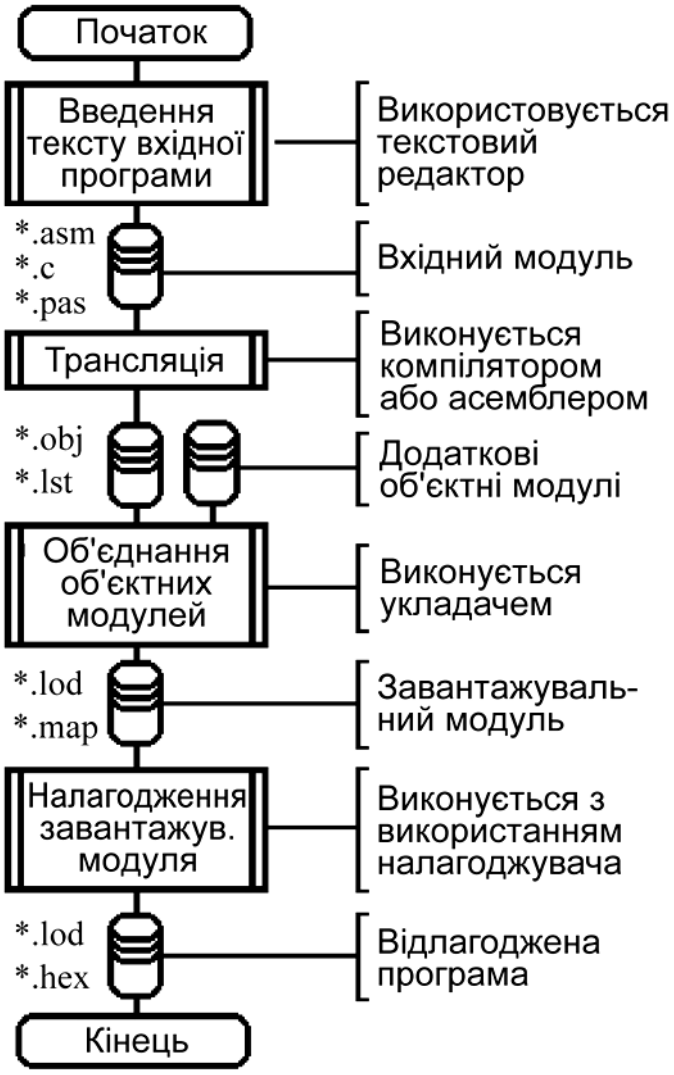


Рис. 3.1. Схема підготовки програм за допомогою інтегрованого інструментального середовища

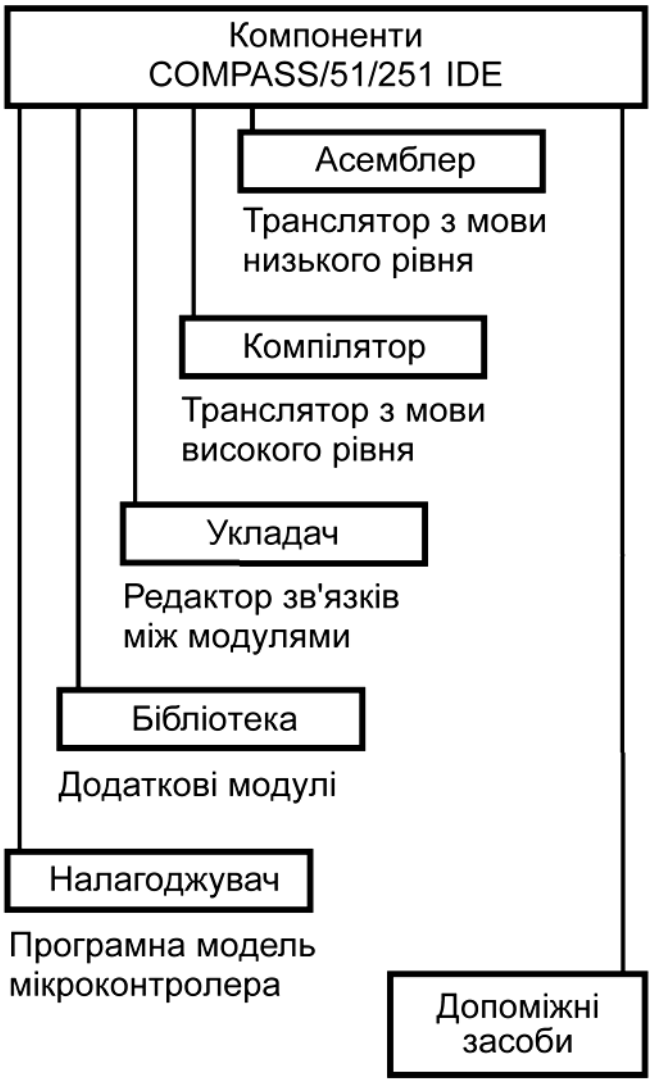


Рис. 3.2. Структура та склад інтегрованого інструментального середовища COMPASS/51/251 IDE

3.2. Введення тексту програми

Початкове введення і редагування тексту програми виконується за допомогою текстового редактора (Text Editor). Текстовий редактор найчастіше не входить в інтегроване інструментальне середовище, а є компонентом системного програмного забезпечення персональної ЕОМ, наприклад, Блокнот або WordPad в операційній системі Windows.

Текстовий файл – це послідовність записів (рядків) із алфавітних (латинські літери), цифрових (0, 1, …F) та управляючих символів. Як розділювач рядків використовується символ "кінець рядка" (EOL, End of Line) або у деяких системах комбінація символів "повернення каретки" (CR) та переведення рядка (LF). Кінець файлу позначається відповідним символом (EOF, End of File).

Кожен рядок початкового текстового файлу – це рядок програми, яка написана в мнемонічних кодах асемблера конкретного мікропроцесора (мікро-

ЕОМ) або на мові високого рівня (C, Pascal, C++, Basic) з дотриманням синтаксису мови і правил програмування. Підготовлений файл зберігається як текстовий, але з розширенням, яке відповідає мові програмування (*.asm – для програми на Асемблері; *.c – для програм на мові С; *.cpp – для програм на мові С++).

3.3. Трансляція текстового файлу

Основна функція програми-транслятора як інструмента інтегрованого інструментального середовища полягає в послідовному перегляді (скануванні) рядків початкової програми (Source File) і перетворенні кожного з них в один або декілька операторів об’єктної програми. Послідовність отриманих таким чином кодів називається об’єктною програмою. При трансляції символьні імена (мнемокоди) або оператори програми на мові високого рівня замінюються еквівалентами в двійковому або гексадецимальному форматі (константи в шістнадцятковому форматі мають позначку H або h, причому, перед ними ставиться символ #). Наприклад, фрагмент програми

```
MOV A, #37H    ; завантаження в регістр A константи 37H
MOV F0, A         ; запис вмісту регістра A в регістр B
```

буде замінений фрагментом об’єктної програми у вигляді послідовно розташованих наступних кодів, а саме: 74, 37, F5 (коди подані у гексадецимальній системі счислення).

Для трансляції асемблерної (написаної в мнемокодах команд мікроконтролера) програми використовують інструмент ІІС – асемблер (Assembler), а для трансляції програми на мові високого рівня – компілятор (Compiler). Результатом роботи трансляторів є об'єктний файл (Object File) і листинговий файл (Listing File). Об'єктний файл у вигляді кодів відтрансльованої програми має розширення *.obj і супроводжується службовою інформацією. Листинговий файл (файл для роздруку) є текстовим за своєю структурою (має розширення *. lst) і містить текст програми у мнемокодах, її шістнадцяткові коди, їх адреси, повідомлення про похибки, попередження тощо. Для перегляду листингового файлу його слід відкрити за допомогою одного із текстових редакторів (наприклад, WordPad, Блокнот, AkelPad), після чого його можна роздрукувати на папері за допомогою принтера. Імена об'єктного і листингового файлів співпадають з ім'ям файлу початкової програми, яку транслювали.

3.4. Компонування об’єктних модулів

Компонування, або об’єднання окремих об’єктних модулів (об’єктних файлів, отриманих транслюванням програм користувача) і модулів із бібліотек, в один модуль виконується за допомогою спеціального редактора зв'язків або компонувача (укладача, Linker). Результатом компонування є завантажувальний модуль (Image File, Load Image File) у вигляді файлу з розширенням *.lod. Ім’я завантажувального модуля за умовчанням співпадає з ім'ям поточного проекту. Компонування необхідне для можливості здійснення зовнішніх звернень одних

об’єктних модулів до інших, оскільки кожна програма користувача транслювалася окремо і незалежно від інших. Тобто, для урахуванням того, що під час трансляції кожної програми інші модулі зі своїми іменами або мітками, їх значеннями або відносними адресами були невідомі.

Якщо в початкових текстових файлах абсолютні адреси модулів не були задані, то завантажувальний модуль буде переміщуваним (тобто, всі адреси команд і переходів будуть відносними і модуль можна завантажити в будь-який сегмент пам'яті програм). Остаточно абсолютні значення відносних адрес задаються і обчислюються під час завантаження скомпонованого модуля в емулятор – електронний пристрій апаратного моделювання роботи мікроконтролера) або в його програмний замінник – симулятор (налагоджувач). В процесі завантаження визначається константа CONST – завантажувальне зміщення, яке додається до всіх відносних адрес. У підсумку абсолютна адреса ABS = REL + CONST, де REL – відносна адреса. Переміщуваність модулів реалізується при дотриманні правил:

– усі чисельні константи є абсолютними;

– лічильник адреси є відносним;

– ім’я у полі позначки є відносним і лише у директиві EQU визначається типом об'єкта в полі операнда (наприклад, визначена виразом M1 EQU 25 позначка є непереміщуваною);

– вирази, які складаються з абсолютних імен та констант є абсолютними;

– різниця двох переміщуваних імен є абсолютною, *наприклад*:

а) якщо M1 та M2 – імена позначок відносних адрес, то число M2 – M1 є абсолютним і не залежить від початкової адреси модуля

```
M1:         JMP     M2
            …
M2:         …
```

б) якщо REL – переміщуване ім'я, ABS – абсолютна константа, то вирази REL, REL ± ABS, ABS ± REL є переміщуваними, наприклад:

```
PC:         JMP     NEXT + 2
            …
NEXT:       …
NEXT + 1    …
NEXT + 2    …
```

У деяких сучасних інтегрованих інструментальних середовищах завантаження в емулятор модуля із розширенням *.lod і визначення абсолютних значень відносних адрес переміщуваних імен виконується компонувачем (укладачем) на останньому етапі компонування. При цьому константа зміщення (фактично – початкова адреса модуля) задається оператором через опцію "конфігурація мішені" (Configure Target), де мішень – програмний (симулятор) або апаратний (емулятор) пристрій, в який завантажується скомпонований модуль (рис. 3.3).

Етап компонування є обов'язковим навіть тоді, якщо програма складається лише з одного модуля.

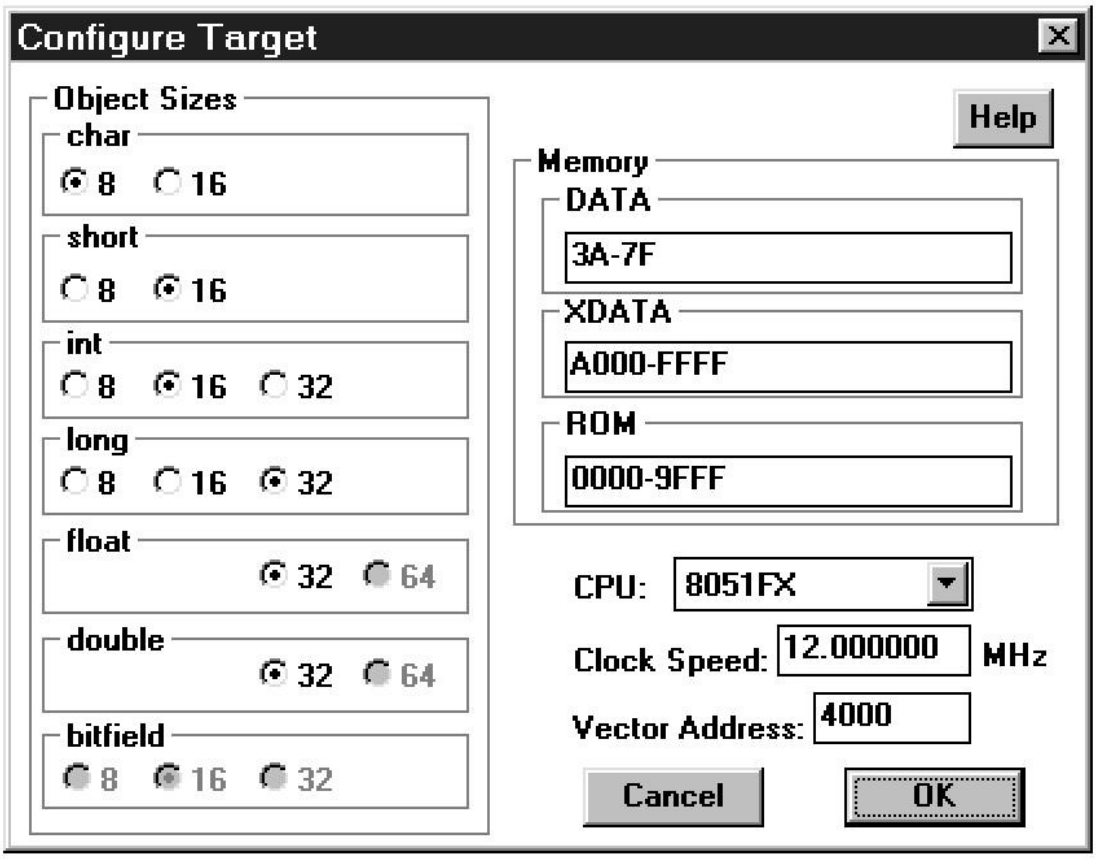


Рис. 3.3. Вигляд вікна вибору конфігурації мішені (Configure Target)

3.5. Налагодження завантажувального модуля

Для локалізації і виправлення логічних похибок, які можуть бути виявлені лише при виконанні програми, застосовують спеціальні засоби емулятори (Emulator) і симулятори (Simulator). Налагодження виконується за допомогою інструмента інтегрованого інструментального середовища – відлагоджувача (Debugger) – програмної оболонки, яка дозволяє користувачу оперативно переглядати і змінювати вміст регістрів і елементів пам'яті даних мікроконтролера або мікропроцесора під час виконання програми завантаженого модуля, тобто дозволяє спостерігати за його роботою зсередини.

Симулятор – це моделююча програма, яка відображає архітектуру модельованої мікро-ЕОМ, мікроконтролера чи мікропроцесора та алгоритми виконання всіх його команд. Після завантаження до налагоджувача і запуску програмного модуля із розширенням *.lod він виконується симулятором саме так, як виконувався б цільовою мікро-ЕОМ, мікропроцесором або мікроконтролером. Користувач взаємодіє з симулятором за допомогою команд налагоджувача, які вводяться з терміналу. Таким чином, по відношенню до налагоджувача (управляючої програмної оболонки) симулятор є драйвером.

Відомі наступні основні способи пошуку логічних помилок:

– дампінг пам'яті – перегляд вмісту елементів пам'яті та регістрів після виконання програми. Для повного виявлення похибок дамп пам'яті прочитують

декілька разів. Недолік способу – необхідність аналізу громіздкої послідовності чисел;

– поетапне виконання програми за допомогою контрольних крапок (Break Point). Програма розбивається контрольними точками на фрагменти, які виконуються послідовно один за іншим із зупинками по закінченню чергового фрагмента. Недолік способу – можливо тільки наближено (у межах контрольованого блоку) локалізувати помилку;

– покрокове (покомандне) виконання програми завантаженого модуля. Недолік способу – значні витрати часу;

– трасування – різновид покрокового способу аналізу роботи програмного модуля. Під час виконання модуля ЕОМ автоматично друкує (або записує у файл трасування) поточні команди і дані, що пересилаються їй.

4. ІНТЕГРОВАНЕ ІНСТРУМЕНТАЛЬНЕ СЕРЕДОВИЩЕ COMPASS/51 IDE

4.1. Загальні відомості

Інтегроване інструментальне середовище COMPASS/51 IDE фірми Production Languages Corporation призначене для розробки та налагодження об'єктно-орієнтованого програмного забезпечення для однокристальних мікро-ЕОМ сімейства MCS-51.

Склад інструментів цього ІІС показаний на рис. 3.2 і містить:

– **асемблер**, або транслятор з мови програмування низького рівня;

– компілятор, або транслятор з мови програмування високого рівня;

– укладач, або редактор зв'язків між модулями, що відтрансльовані;

– бібліотека із набору допоміжних програмних модулів;

– налагоджувач, який застосовується при відлагодженні програми.

Для розробки мікропроцесорних систем на основі мікроконтролерів сімейства MCS-251 доцільно користуватися таким інтегрованим інструментальним середовищем як COMPASS/251, можливості і оформлення якого схожі з ІІС COMPASS/51.

4.2. Інсталяція COMPASS/51 IDE та підготовка до роботи

Інсталяція починається запуском файлу Setup.exe з ліцензійної дистрибутивної дискети або CD-ROM. Під час інсталяції слід дотримуватися інструкцій, що виводяться на екран. Після інсталяції до папки …\PB51IDE\BIN необхідно записати програму симулятора 51sim.exe (у випадку, якщо відсутня плата внутрішньосхемного емулятора) або драйвер підключеного до ЕОМ емулятора.

4.3 Запуск COMPASS/51 IDE і підготовка проекту

COMPASS/51 IDE запускається послідовним виконанням програм Пуск/ Програми/… Після запуску на екрані з'являється головне вікно інтегрального інструментального середовища і запрошення до роботи. Для роботи з ІІС слід натиснути кнопку "Skip", для виведення довідки - кнопку "Run COMPASS Quick

Tutorial". Як і будь-який інший Windows-додаток, COMPASS/51 IDE містить панель випадних меню, панель інструментів і робочий простір. У панелі інструментів розташовані найчастіше використовувані команди і інструменти даної ІІС, повний же набір команд, згрупованих за схожими ознаками, знаходиться в панелі меню, які випадають. Головне вікно ІІС містить рядок меню команд та панель інструментів (рис. 4.1). На рис. 4.1 у рядку меню показані:

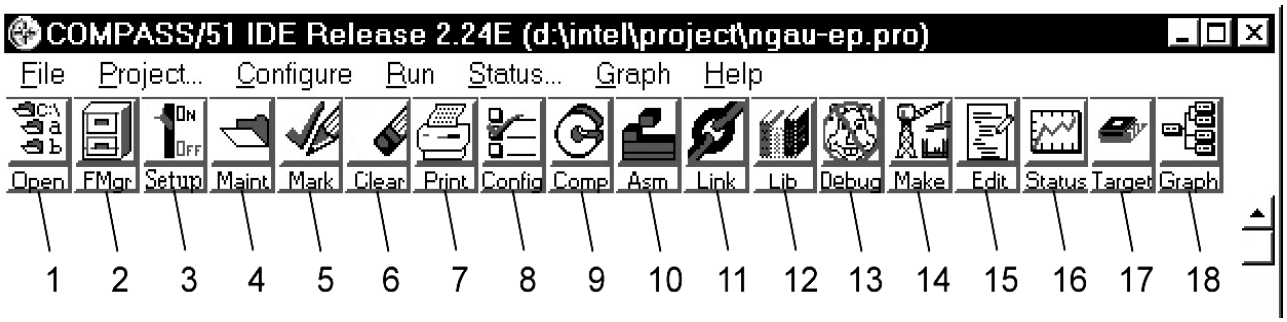


Рис. 4.1. Головне вікно (рядок меню та панель інструментів) інтегрованого інструментального середовища COMPASS/51/251 IDE

- Команди групи **File** призначені для виконання операцій створення, збереження, завантаження, редагування і виведення на друк файлів, які створені в оболонці COMPASS/51, а також поточної інформації щодо проекту:
  New Project… – створити файл нового проекту;
  Open Project… – відкрити файл створеного раніше проекту;
  Save Project… – зберегти останні зміни поточного проекту;
  Save As Project… – зберегти поточний проект під заданим ім'ям;
  Edit File… – переглянути і відредагувати файл за допомогою зіставленого з його розширенням текстового редактора;
  Print File… – вивести файл на друкування у текстовому вигляді;
  Print Window… – друкувати вміст головного вікна;
  Clear – очистити головне вікно;
  Mark – відобразити поточний час і дату у головному вікні;
  Exit – вихід з інтегрованого інструментального середовища COMPASS/51.

- Команда **Project…** – відкриває вікно складу проекту Project Maintenance, яке призначене для призначення зв'язків окремих файлів, що зберігаються на жорсткому диску, з поточним активним проектом (рис. 4.2).

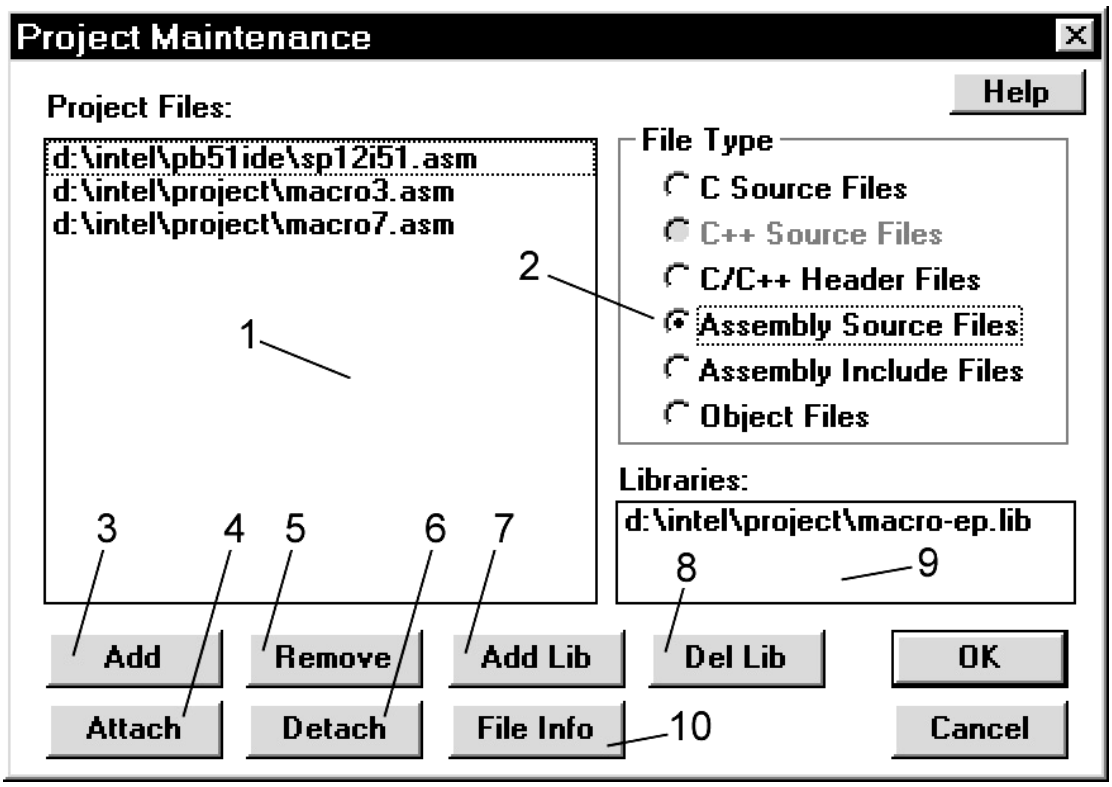


Рис. 4.2. Вікно керування поточним проектом (Project Maintenance)

Діалогове вікно **Project Maintenance** складається з вікна 1 списку файлів проекту (**Project Files**), перемикача типів файлів 2 (**File Type**), кнопки 3 (**Add**) додавання файлу до поточного проекту, кнопки 4 (**Attach**) додавання файлу до вибраної бібліотеки, кнопки 5 (**Remove**) видалення файлу з поточного проекту, кнопки 6 (**Detach**) від'єднання файлу від вибраної бібліотеки; кнопка 7 (**Add Lib**) додавання бібліотеки до проекту; кнопка 8 (**Del Lib**) видалення вибраної бібліотеки із проекту; вікно 9 списку підключених бібліотек проекту; кнопка 10 (**File Info**) отримання інформації про вибраний файл.

- Команди групи **Configure** (команди вибору конфігурації):
  Assembler… – асемблера;
  Compiler… – компілятора;
  Debugger… – налагоджувача;
  Librarian… – бібліотеки;
  Linker… – укладача;
  Optimizer… – оптимізатора функцій компіляції програм;
  Target… – мішені;
  Environment… – середовища COMPASS/51 IDE.

  Дана група команд призначена для гнучкої настройки користувачем, яка є у складі інструментальної бази COMPASS/51 IDE.

- Команди групи **Run (**команди запуску інструментів IIC COMPASS/51):
  Assembler… – асемблера;
  Compiler… – компілятора;
  Debugger… – відлагоджувача;

Librarian… – бібліотеки;
Linker… – укладача;
Make… – інструмента "виконати проект".

- Команда **Status**… – відкриває вікно статусу (стану) задач.

- Команди групи **Graph** (команди виклику вікон графів):
  Calls – граф залежностей викликів;
  Files – граф залежностей файлів.

- Команди групи **Help**, команди виклику довідки.

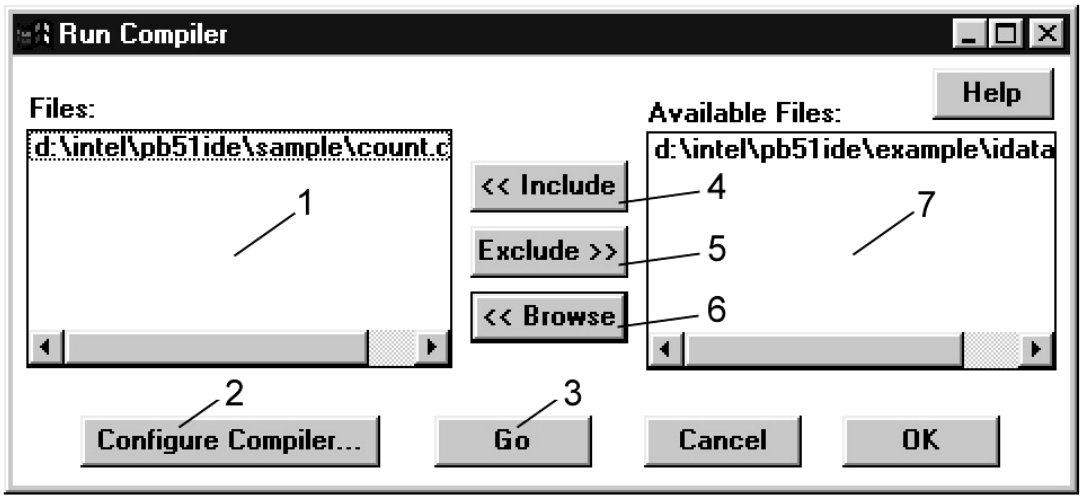
На рис. 4.1 додатково показані кнопки панелі інструментів, більшість з яких дублює відповідні функції команд рядка головного меню:

1 – **Open** – відкрити проект;
2 – **File Manager** – запуск диспетчера файлів;
3 – **Setup** – вибір конфігурації середовища;
4 – **Maintenance** – відкрити вікно Project Maintenance складу проекту;
5 – **Mark** – вивести поточний час і дату у головному вікні;
6 – **Clear** – очистити головне вікно;
7 – **Print** – друкувати текст з головного вікна;
8 – **Configure** – вибір конфігурації інструменту;
9 – **Compiler** – запуск компілятора;
10 – **Assembler** – запуск асемблера;
11 – **Linker** – запуск редактора зв'язків (укладача);
12 – **Librarian** – запуск бібліотеки;
13 – **Debugger** – запуск налагоджувача;
14 – **Make** – виконати проект;
15 – **Edit** – редагувати файл зіставленим його розширенню текстовим редактором;
16 – **Status** – відкрити вікно стану задач;
17 – **Target** – вибрати конфігурацію мішені;
18 – **Graph** – показати граф залежностей.

Після запуску COMPASS/51 IDE необхідно за допомогою використання команди **New Project…** головного меню створити новий проект – сукупність файлів з якими передбачається працювати. Для спрощення роботи над проектом бажано підключити до нього початкові (вхідні) файли, які розроблені програмістом-користувачем. Файли, які створюються інтегральним інструментальним середовищем, автоматично зберігатимуться у папці, в якій знаходиться головний файл проекту *.pro.

Для опису проекту (підключення до нього певних файлів) використовується вікно **Project Maintenance** (див. рис. 4.2). Обравши тип файлу за допомогою кнопки **File Type**, його можна добавити в проект (**Add**), видалити з проекту (**Remove**), приєднати до бібліотеки (**Attach**), від'єднати від бібліотеки (**Detach**), одержати інформацію про файл (**File Info**). При асемблюванні або компілюванні

файли, які підключені до проекту, автоматично з'являються у вікнах доступних файлів (**Available Files**) проекту (рис. 4.3), що звільняє користувача від їх пошуку за допомогою кнопки **Browse**.



1 - Вікно компілюємих файлів (Files);
2 - Кнопка вибору конфігурації компілятора (Configure Compiler);
3 - Кнопка запуску процесу компіляції (Go);
4 - Кнопка включення файлу до складу компілюємих (Include);
5 - Кнопка виключення файлу із складу компілюємих (Exclude);
6 - Кнопка пошуку файлів (Browse);
7 - Вікно доступних у проекті файлів (Available Files)

Рис. 4.3. Вікно запуску компілятора 51C Compiler інтегрованого інструментального середовища COMPASS/51

Підключені до проекту файли можна проглядати і редагувати текстовим редактором, який запускається кнопкою **Edit** (см. рис. 4.1) або командою **Edit File** меню команд головного вікна. В результаті на екрані з’являється вікно **Edit File** (рис. 4.4) для вибору необхідного файлу.

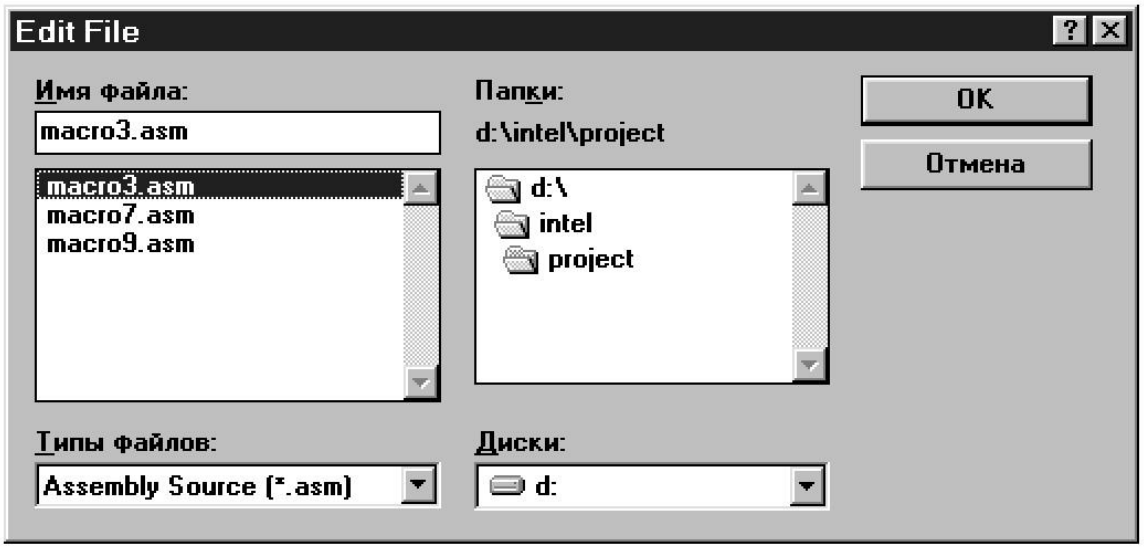


Рис. 4.4. Вікно запуску файлу на редагування

Для зіставлення типу файлу і програми текстового редактора використовуються засоби операційної системи або диспетчер файлів (рис. 4.5), який викликається кнопкою **File Manager**.

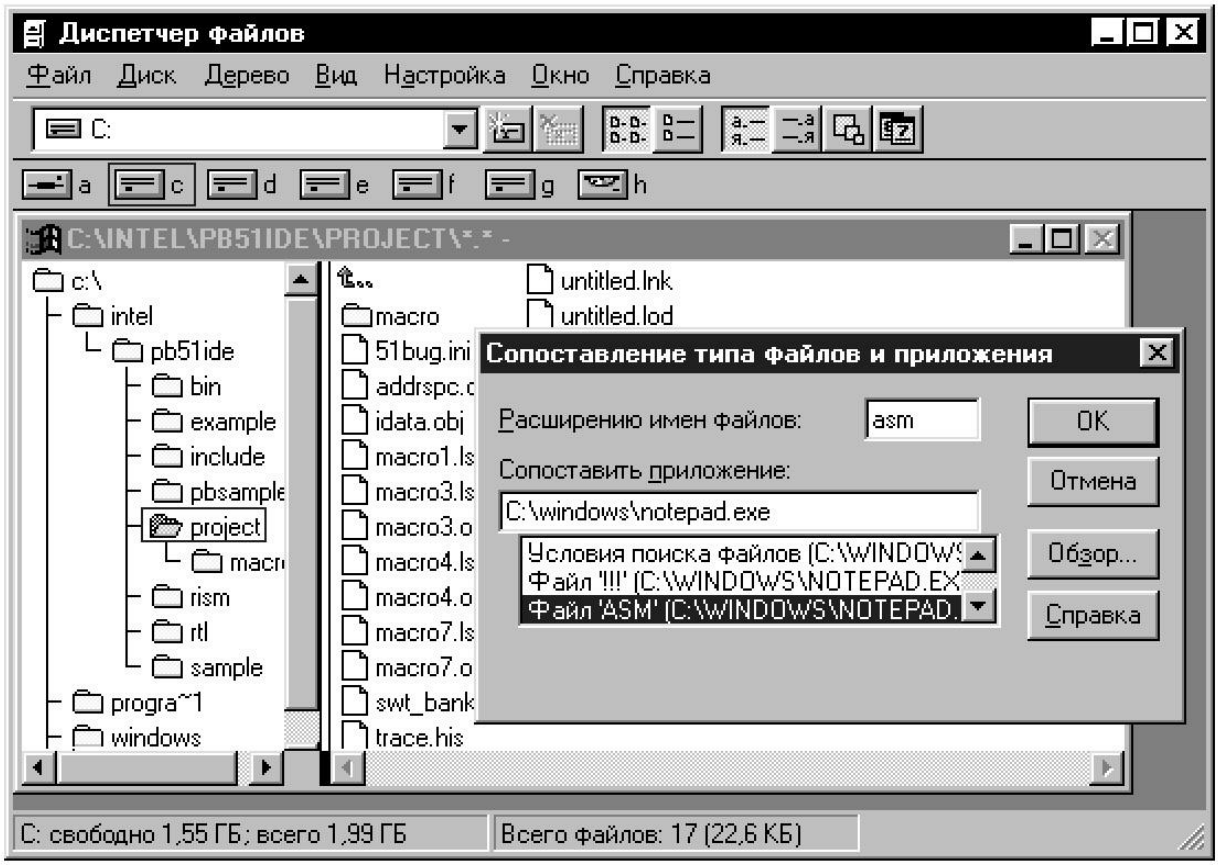


Рис. 4.5. Вікно диспетчера файлів

4.4. Трансляція і компонування модулів

Для трансляції файлів програм (початкових, вхідних модулів) на мові C/C++ користуються компілятором, який запускається кнопкою **Compiler** панелі інструментів головного вікна (див. рис. 4.1) або командою **Compiler** меню **Run**. Після цього файли, які необхідно транслювати, кнопкою **Include** переносять із списку доступних в проекті файлів (поз. 7 на рис. 4.3) до списку файлів, які транслюються (див. поз. 1 на рис. 4.3). Якщо ж необхідно виконати трансляцію файлу, який не підключений до даного проекту, то цей файл шукають за допомогою кнопки **Browse** (див. поз.6 на рис. 4.3). При необхідності, за допомогою кнопки **Exclude** (див. поз. 5 на рис. 4.3), непотрібні файли можуть бути легко видалені із списку файлів для трансляції.

Запуск компіляції вибраних файлів виконують кнопкою **Go** (див. поз. 3 на рис. 4.3). Для зміни властивостей компілятора передбачена кнопка **Configure Compiler** (див. поз. 2 на рис. 4.3).

Для запуску асемблера потрібно виконати аналогічні дії. Але перед асемблюванням програми у вікні **Configure Assembler** (рис. 4.6), яке викликається кнопкою вікна **Run Assembler** або з головного вікна, необхідно звернути увагу на наступні опції:

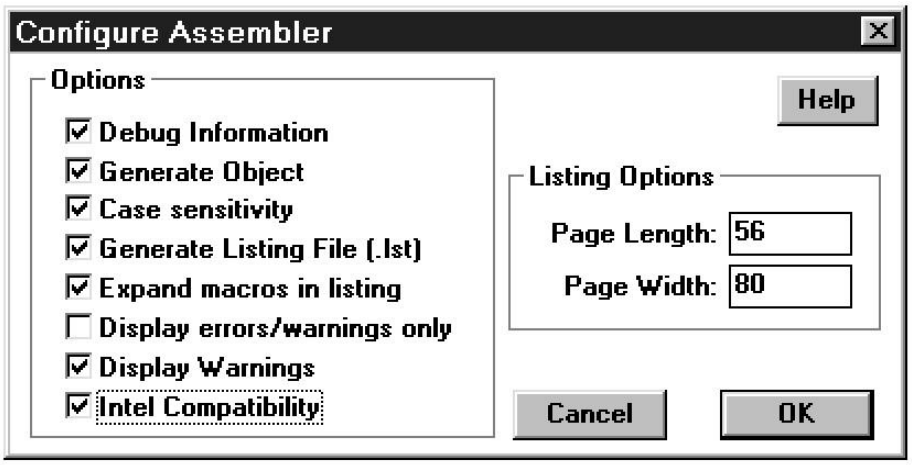


Рис. 4.6. Вікно конфігурації асемблера

- **Debug Information** – генерувати інформацію для вікна проекту налагоджувача;
- **Generate Object** – генерувати (створювати) об'єктний файл;
- **Case Sensitivity** – розрізняти великі та малі літери в іменах змінних та позначок;
- **Generate Listing File (*.lst)** – генерувати файл лістингу;
- **Expand macros in listing** – друкувати розгорнені макророзширення в лістингу;
- **Display errors/warnings only** – при асемблюванні відображати в головному вікні лише повідомлення про помилки і попередження;
- **Display Warnings** – відображати в головному вікні попередження;
- **Intel Compatibility** – режим Intel - сумісності (у випадку, якщо опція встановлена, чисельні дані не повинні починатися з букви). Не позначені буквами "h" (hex, hexidecimal) або "b" (bin, binary) дані сприймаються як описані в десятковій системі числення.

У меню **Listing Options** вікна **Configure Assembler** можна встановити кількість рядків однієї сторінки лістингового файлу (**Page Length**) і максимальну кількість символів одного рядки (**Page Width**).

Початковій текстовій програмі потрібно надати наступну структуру:

```
Поле мітки: Код операції_Поле операндів; Поле коментарів
```

Приклад:

```
Label1: MOV A,B ; завантаження лічильника
```

Будь-яке з вказаних полів може бути відсутнім, наприклад:

```
Поле мітки      Код операції   Поле операндів   Поле коментарів
                                                ;Приклад програми з
                                                ;незаповненими полями
                INC            DPTR
TIME1:                                          ;п/п затримки часу
M1:             MOV            R5, #0F3h        ;завантаження лічильника
                NOP
                DJNZ           R5, M1
```

Операнди обов’язково відокремлюються від коду операції одним або декількома символами пропуску або табуляції (→|). При підготовці тексту програми до роздрукування для зручності бажано всі поля відділяти символами табуляції. Мітка відділяється від коду операції (інструкції) двокрапкою (**:**), а поле коментаря – крапкою з комою (**;**).

Компонування об'єктних модулів виконується за допомогою програми укладача, яка запускається з головного вікна інтегрованого інструментального середовища кнопкою **Link** (див. рис. 4.1) або командою **Run Linker…**. Після цього на екрані монітора з’являється вікно запуску укладача **Run Linker** (рис. 4.7), в якому доступні наступні процедури:

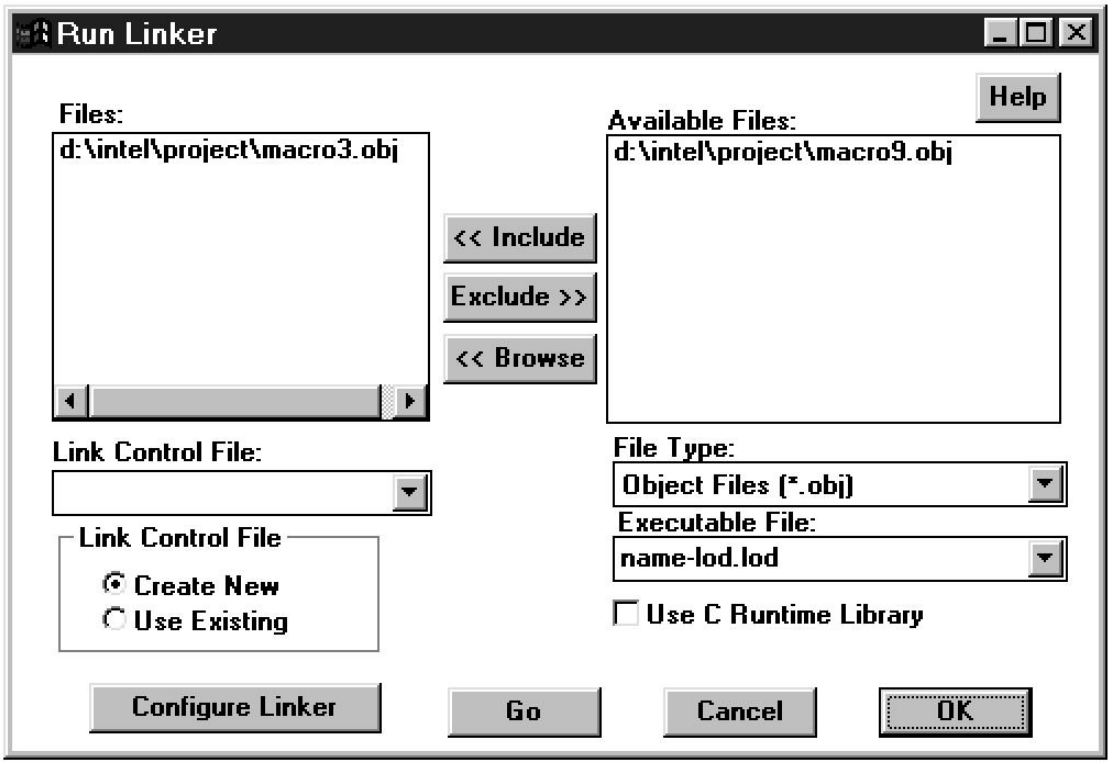


Рис. 4.7. Вікно запуску укладача

- **Link Control File:** – ім'я файлу, в якому буде збережена інформація щодо наступного сеансу роботи укладача (за умовчанням ім'я файлу *.lnk співпадатиме з ім'ям поточного проекту *.pro);
- **Create New** – створити новий файл компонування *.lnk;
- **Use Existing** – використовувати вже існуючий файл компонування *.lnk, ім'я якого введене до рядка Link Control File (фактично при цьому не виконується компонування файлів, а об'єктні модулі, які відображені в списку Files вікна Run Linker, будуть розташовані в завантажувальному модулі згідно існуючій карті компонування);
- **File Type:** – вказати тип файлів, що підлягають компонуванню (*.obj – об'єктний, *.lib – бібліотечний модуль);
- **Executable File:** – вказати ім'я завантажувального модуля, який буде створений укладачем, із розширенням *.lod (за умовчанням ім’я модуля співпаде з ім'ям поточного проекту із розширенням *.pro);

- **Use C Runtime Library** – використовувати бібліотечний модуль автозапуску (за звичай цю опцію вимикають).

У разі потреби перед запуском процесу компонування кнопкою **Go** можна задати конфігурацію укладача, для чого кнопкою **Configure Linker** вікна **Run Linker** або з головного вікна IIC (див. поз. 8 на рис. 4.1) слід викликати однойменне вікно **Configure Linker** (рис. 4.8). Завдяки цьому стануть доступними наступні опції:

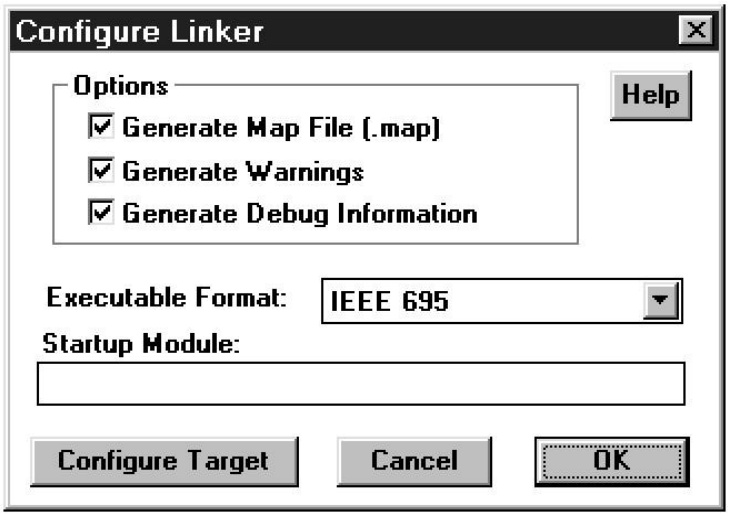


Рис. 4.8. Вікно конфігурації укладача

- **Generate Map File (*.map)** – генерувати файл-карту розподілу ад-ресного простору між початковими модулями (*.obj, *.lib);
- **Generate Warnings** – відображати в головному вікні інтегрованого інструментального середовища попередження укладача;
- **Generate Debug Information** – генерувати інформацію для вікна проекту налагоджувача (при цьому використовується відповідна інформація Debug Information, яка отримана при асемблюванні початкових текстових файлів, див. рис. 4.8);
- **Startup Module:** – задати ім'я і шлях до файлу стартового модуля (С - програми) у разі його наявності;
- **Executable Format:** – вибрати формат завантажувального модуля:
  - IEEE 695;
  - Intel Hex-Records,
  - Intel OMF51,
  - Motorola S-Records,

де формат **IEEE 695** відповідає міжнародному технічному стандарту, а решта – стандартам відповідних фірм (вибраний стандарт повинен забезпечити сумісність завантажувального модуля з платою емулятора або програматора).

Для встановлення розрядності операндів різного типу, адресного простору даних і програми, типу мікро контролера і його тактової частоти, а також адресного простору векторів переривання слід викликати на екран вікно **Configure Target** (рис. 4.9). Для цього у вікні **Configure Linker** слід натиснути

кнопку **Configure Target** (або у головному вікні IIC натиснути **Config** чи вибрати команду **Target** з випадаючого меню **Configure**).

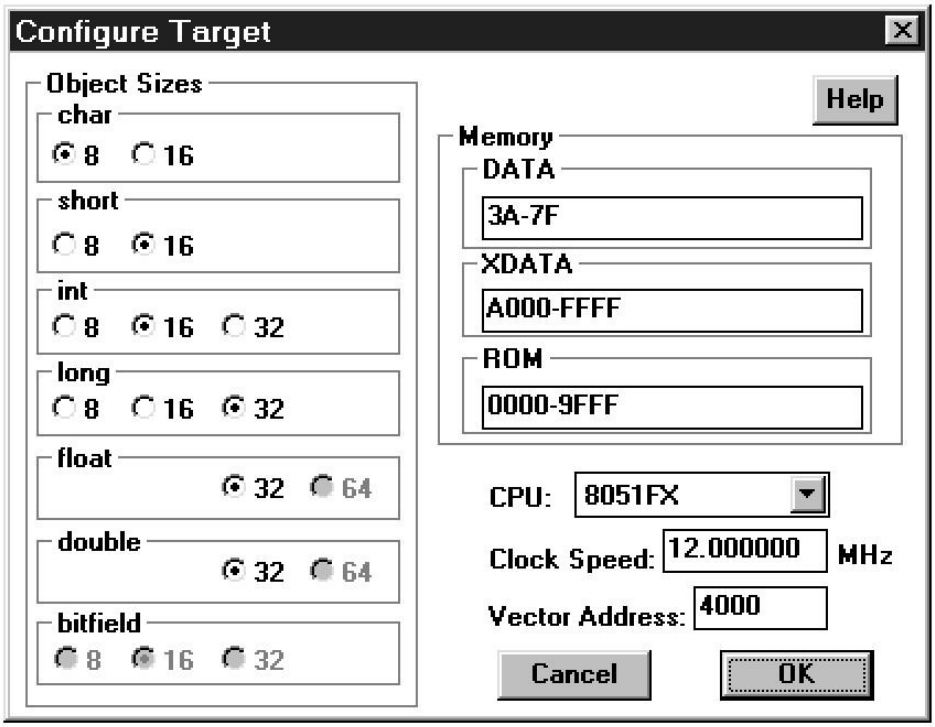


Рис. 4.9. Вікно конфігурації мішені

Завдяки цьому стають доступними такі можливості:

- **Object Sizes** – встановити розрядність типів даних вихідної програми, для якої необхідно отримати об’єктний модуль;
- **Memory** – задати адреси елементів пам'яті даних (DATA), зовнішньої пам'яті даних (XDATA) та пам'яті програм (ROM). За відсутності визначення в текстових файлах (наприклад, директивою ORG) початкової адреси програми, коди програми із завантажувального модуля із розширенням *.lod розміщуються в ПЗП з початкової адреси, яку слід вказати в рядку ROM;
- **CPU:** – вибрати необхідний тип мікроконтролера з наступного переліку: 8031, 8032, 8051, 8051FX, 8051GB, 8052;
- **Clock Speed:** – вказати тактову частоту мікроконтролера (Мгц).

Якщо за допомогою вікна **Project Maintenance** (див. рис. 4.2) до проекту приєднані всі необхідні файли, то операції транслювання і компонування приєднаних модулів можна виконати за допомогою кнопки **Make** головного вікна IIC. Початкові файли опрацьовуються програмами в полях інструментів (Tool Names) вікна **Run Make** (51cc.exe, 51asm.exe, 51link.exe, 51lib.exe), наведеного на рис. 4.10). При цьому доступні такі можливості:

- **Build Project** – створити проект;
- **Create Make File** – згенерувати файл, в якому будуть збережені поточні параметри інструмента Make, який потім можна запускати за допомогою кнопки Run Make File;
- **Run Linker** – запускати укладач при виконанні проекту;

- **Use C Runtime** – використати модуль автозапуску.

Як і при використанні укладача опцію **Use C Runtime**, як правило, слід вимкнути.

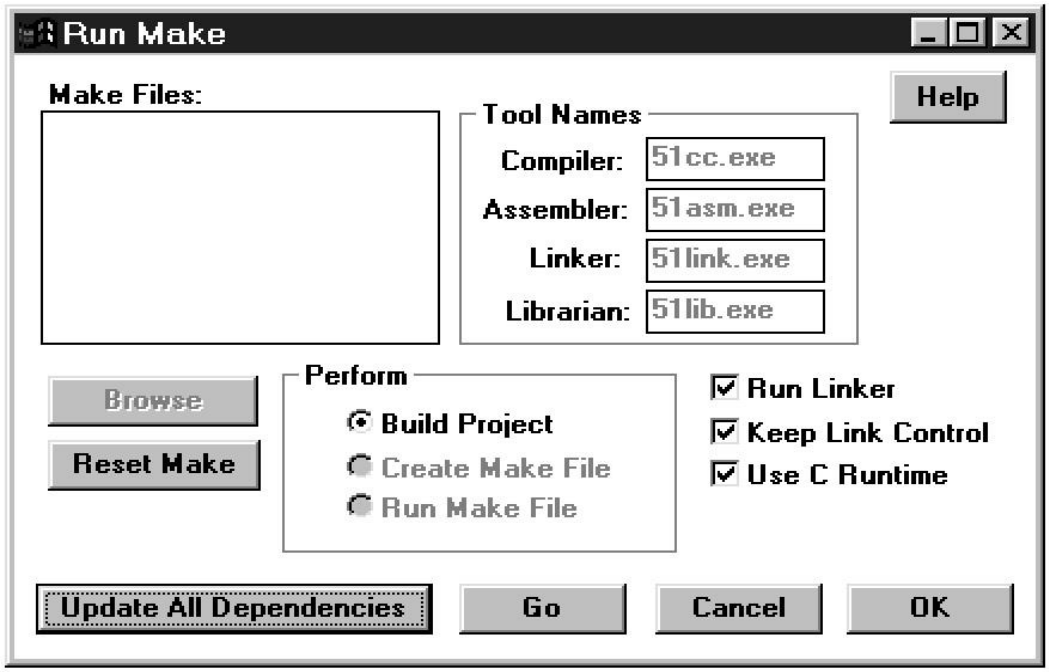


Рис. 4.10. Вікно запуску повного циклу обробки проекту

4.5. Робота з налагоджувачем

Перевірити і налагодити роботу завантажувального модуля можна за допомогою налагоджувача, який викликають натисканням кнопки **Debugger** (поз. 13 на рис. 4.1) або командою **Run Debugger** з випадаючого меню **Run** головного вікна інтегрованого інструментального середовища. Заздалегідь слід задати конфігурацію налагоджувача за допомогою можливостей опцій вікна **Configure Debugger** (рис. 4.11).

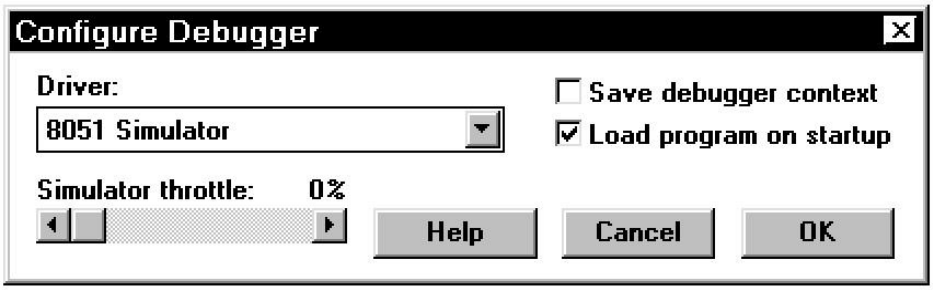


Рис. 4.11. Вікно конфігурації налагоджувача

Вікно **Configure Debugger** надає наступні можливості:

- **Driver:** – вибрати драйвер симулятора або емулятора, який буде використано налагоджувачем;
- **Save debugger context** – зберігати вигляд головного і допоміжних вікон налагоджувача;

- **Load program on sturtup** – читати завантажувальний модуль *.lod при запуску налагоджувача;
- **Simulator throttle** – встановити швидкість моделювання налагоджувачем роботи мікроконтролера, який емулюється.

Головне вікно (рис. 4.12) налагоджувача містить панель робочих кнопок та вікно індикації стану виконання завантаженого модуля у правому верхньому кутку (наприклад, під час виконання програми висвітлюється напис RUN). У інших випадках написи TIMED, ANIMATE або STOP). Окрім того, у головному вікні розташовані додаткові вікна (у згорненому або розгорненому стані):

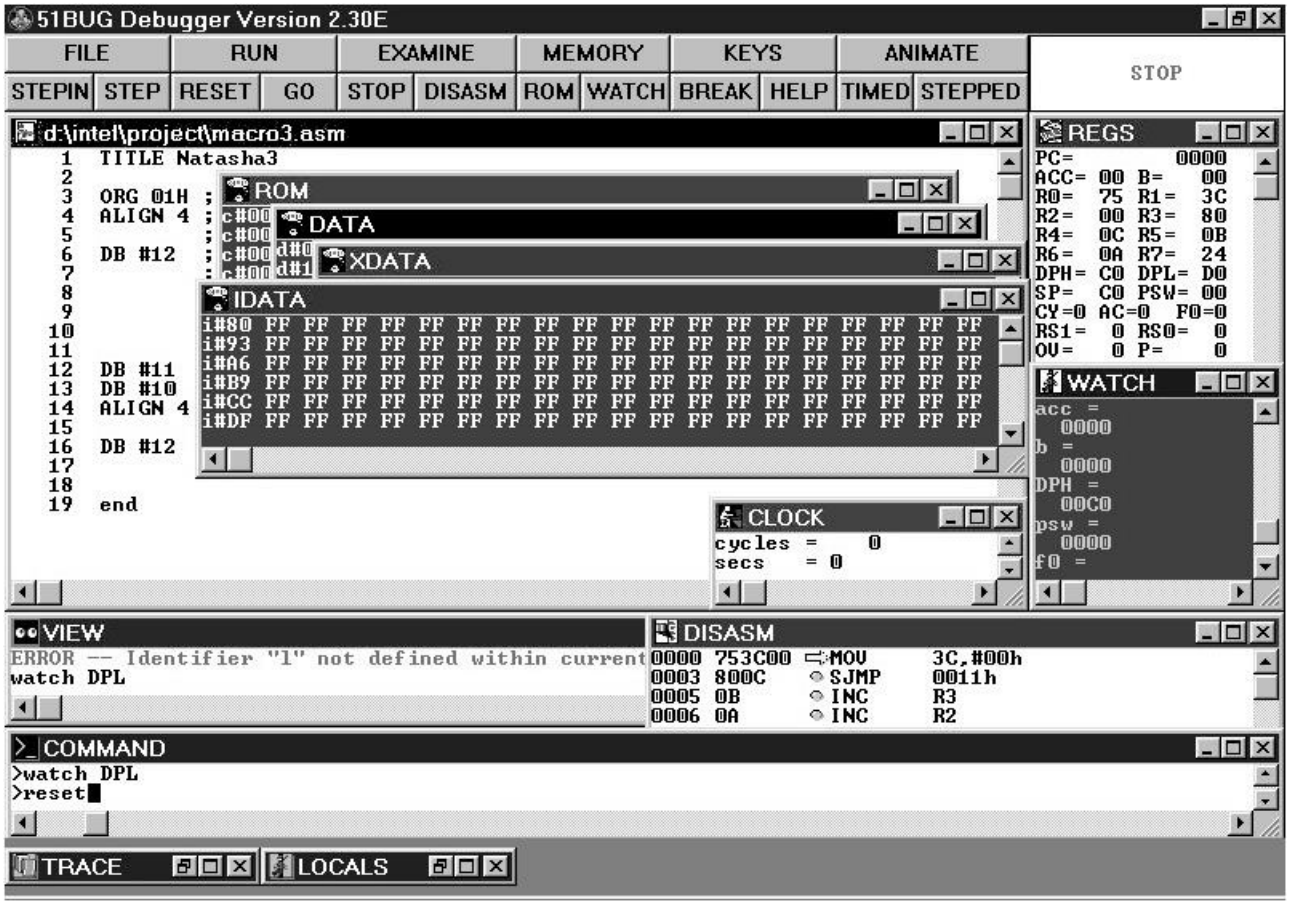


Рис. 4.12. Головне вікно налагоджувача

- **CODE** – вікно відображення вихідного тексту активного (поточного) модуля проекту. За його відсутності у рядку заголовка вікна замість шляху до першого файлу текста програми (наприклад, d:\intel\progect\ macro3.asm) висвітлюється назва вікна **CODE** з написом у ньому No files loaded;
- **REGS** – вікно вмісту регістрів мікроконтролера;
- **ROM** – вікно вмісту пам'яті програм;
- **DATA** – вікно вмісту елементів пам'яті;
- **IDATA** – вікно вмісту внутрішньої пам'яті даних мікроконтролера;
- **XDATA** – вікно вмісту зовнішньої пам'яті даних;
- **CLOCK** – вікно індикації кількості виконаних машинних циклів і поточного часу виконання мікроконтролером завантаженого модуля;

- **WATCH** – вікно відображення вмісту вибраних користувачем регістрів і елементів пам'яті, звернення до яких здійснюється через визначені в програмі ідентифікатори;
- **VIEW** – вікно виведення діалогової інформації при виконанні директив командного рядка;
- **DISASM** – вікно дизасемблера (відновлює асемблерну програму за кодами її команд);
- **COMMAND** – вікно для введення команд налагоджувача;
- **TRACE** – вікно трасування програми, в якому відображається інформація про вміст основних регістрів і переміщення даних між ними після кожної виконаної операції;
- **LOCALS** – вікно індикації локальних змінних, функцій і їх значень.
- Робоча панель вікна налагоджувача має декілька інструментів.
- Команди випадного меню кнопки **FILE**:

Browse File – знайти файл і відкрити його в окремому вікні налагоджувача;
List Modules – вивести у вікні **VIEW** адреси і кількість зайнятих байт сегментів пам'яті для всіх вхідних модулів, які увійшли до кінцевого завантажувального модуля;
Switch Files – підключити файл, тобто вивести текст програми файлу із розширенням *.asm у вікно **CODE**;
Execute Batch File – виконати пакетний файл із розширенням *.bat. Команди у цьому файлі повинні відповідати синтаксису інструкцій налагоджувача;
Exit – вийти з налагоджувача.

- **STEPIN** – виконати одну поточну команду асемблерної програми;
- **STEP** – виконати одну поточну підпрограму, команди тіла одного циклу або одну команду, яка не належить до підпрограми або циклу;
- **RESET** – імітувати скидання мікроконтролера;
- **GO** – запустити на виконання завантажений модуль;
- **STOP** – припинити виконання завантаженого модуля.
- Команди випадного меню кнопки **RUN**:

Breake Point – викликати меню команд встановлення контрольних точок;
Step Into – команда, рівнозначна **STEPIN**;
Step, **Go**, **Reset** – дії команд аналогічні відповідним кнопкам;

- Команди випадного меню **EXAMINE**:

Examine Binary – перевести в двійковий код введене десяткове число або математичний вираз, який складається з чисел, змінних і функцій С-модуля (результат відображається у вікні **VIEW**);
Examine Decimal – визначити десятковий еквівалент введеного виразу або числа (результат відображається у вікні **VIEW**);
Examine Hexidecimal – визначити шістнадцятковий еквівалент числа або результату обчислення введеного виразу (результат відображається у вікні **VIEW**);
Examine String – вивести у вікні **VIEW** рядок символів, які відповідають числу або результату введеного виразу в системі кодової таблиці ASCII.

- **DISASM** – розгорнути/згорнути вікно дизасемблера;
- **WATCH** – розгорнути /згорнути вікно **WATCH**;
- Команди випадного меню кнопки **MEMORY**:

Data Type – вибрати формат представлення даних у вікнах **DATA**, **IDATA**, **XDATA**, **ROM** (**Character** – по одному байту, **Short** – 2 байти, **Long** – 4 байти, **Integer** – ціле число, **Float** – число з плаваючою комою звичної точності, **Double** – число з плаваючою комою подвійної точності, **ASCII** – символьний еквівалент числа);

Radix – вибрати систему числення, в якій виводяться дані у вікнах **DATA**, **IDATA**, **XDATA** і **ROM** (**Decimal**, **Hexidecimal**, **Octal**, **Binary** – відповідно десяткова, шістнадцяткова, вісімкова, двійкова).

- Команди випадного меню кнопки **BREAK**:

Toggle Permanent – встановити постійну контрольну точку за адресою команди, яка заздалегідь позначена курсором;

Toggle Temporary – встановити тимчасову контрольну точку;

List BP's – вивести у вікні **VIEW** перелік встановлених контрольних крапок;

Cancel All BP's – вилучити всі контрольні точки.

- **KEYS** – викликати випадне меню з переліком функціональних клавіш (F1 – F9);
- **TIMED** – увімкнути/вимкнути режим часової анімації;
- **STEPPED** – увімкнути/вимкнути режим покрокової анімації;
- **ANIMATE** – задати часовий інтервал анімації.

Функціональні клавіші дублюють призначення кнопок робочої панелі, за винятком F6 – **Window** (зміна активних вікон).

В режимі внутрішньовіконного редагування можна оперативно змінювати інформацію у вікнах **REGS**, **ROM**, **DATA**, **IDATA**, **XDATA**. Для цього слід подвійним натисканням лівої кнопки миші або за допомогою клавіші Enter виділити потрібний байт, ввести його нове значення і натиснути Enter або ліву кнопку миші (виділений байт позначається червоним кольором). Редагувати текст програми у мнемонічних кодах можна аналогічним способом, змінюючи команди у вікні дизасемблера **DISASM** (важливо пам'ятати, що при цьому не змінюються ні об’єктні модулі, ні завантажувальний модуль).

Для встановлення контрольної точки слід розмістити курсор на відповідній асемблерній команді у вікні **CODE** або **DISASM** і кнопкою **BREAKE** задати тип контрольної точки. Постійну контрольну точку можна встановити і подвійним натисканням лівої, а тимчасову – правої кнопки миші на відповідній асемблерній команді.

Для занесення вибраних регістрів або елементів пам'яті у вікно **WATCH** у вікні **COMMAND** необхідно набрати і виконати команду **watch<identifier>** , де **identifier** – ім’я вибраного елементу. Наприклад, **watch ACC**, **watch m1**, **watch R2.**

Зворотну дію виконує команда вилучення елементу з вікна **WATCH**:

```
del[ete] watch <identifier>  .
```

Символи в квадратних дужках є необов'язковими.

Робота з налагоджувачем може проводитися і у діалоговому режимі за допомогою введення команд у вікні **COMMAND** (командний рядок). Повідомлення щодо реакції системи на виконані команди виводяться у вікні **VIEW**. Більшість команд дублює відповідні кнопки робочої панелі налагоджувача.

Найчастіше користуються наступними командами:

- **g[o]** – запустити завантажений модуль на виконання;
- **step** – виконати одну поточну підпрограму, команди тіла одного циклу або одну команду, яка не належить до підпрограми або циклу;
- **stepin** – виконати одну інструкцію асемблерної програми (можливий наступний формат команди: **si[<count>]** , де **count** – кількість виконуваних інструкцій;
- **reset** – імітувати скидання мікроконтролера;
- **q[uit]** – вийти з налагоджувача.

Стежити і керувати роботою інструментів інтегрованого інструменттального середовища можна за допомогою вікна статусу задач **Status**, яке викликається однойменною кнопкою (див. поз. 16 на рис. 4.1) або командою з головного вікна IIC. При цьому стають доступними такі можливості:

- **All** – показувати у вікні Task стан виконання всіх завдань;
- **Active** – показувати стан виконання активних завдань;
- **Kill** – перервати і завершити виконання вибраного завдання;
- **Suspend** – припинити виконання завдання;
- **Resume** – поновити виконання завдання.

4.6. Особливості практичного використання IIC

Для ознайомлення з особливостями практичного використання можливостей IIC спочатку слід створити власну папку, довжина імені якої не повинна перевищувати вісім символів. Ім'я папки (директорії) повинне складатися лише з символів латинського алфавіту, арабських цифр і знаку підкреслення «_». У створену папку необхідно скопіювати файли асемблерних програм, які увійдуть до складу завантажувального модуля (наприклад, файли prog1.asm, prog2.asm та prog3.asm вказані викладачем). Далі потрібно виконати подальші дії з метою отримання завантажувального модуля у наведеній нижче послідовності. Назва проекту може співпадати з прізвищем студента (наприклад, Ivanov.pro).

Послідовність виконання роботи

1. Визначитися із місцем розташування майбутнього проекту (диском, а за необхідності – і папкою на цьому дискові) і створити у цьому місці файл проекту із розширенням *.pro (наприклад, Ivanov.pro). Для досягнення цієї мети слід:

- у випадному меню **File** основного вікна COMPASS/51 вибрати пункт **New Project**;
- у відкритому діалоговому вікні **New Project** для створення нового проекту спочатку у випадному рядку «Диски:» слід вибрати диск (наприклад, e:), а потім вказати шлях до місця розташування файлу у полі «Папки»;

- у рядку «Имя файла» слід ввести ім’я майбутнього проекту (у прикладі Ivanov.pro) і натиснути кнопку Ok.

2. Створити файл асемблерної програми (або послідовно – декілька файлів), для чого необхідно виконати такі дії:

- відкрити діалогове вікно **Edit File** редагування файлів, натиснувши кнопку **Edit** основної панелі інструментів (або вибравши пункт **Edit File** випадного меню **File**);
- задати ім’я файлу програми (наприклад, prog0.asm) і місце його розташування (послідовність дій – як у п. 2). Бажано розміщувати цей текстовий файл разом з файлом нового проекту;
- у випадному рядку «Типы файлов» вибрати тип Assembly Sourse із розширенням *.asm (файл програми на мові асемблера) і натиснути кнопку Ok. У відповідь на повідомлення про відсутність файлу слід відповісти натисканням кнопки «Да» для появи вікна текстового редактора «Блокнот» з ім’ям створеного файлу PROG0 у заголовку;
- у відкритому вікні текстового редактора «Блокнот» розмістити необхідний текст на мові асемблера (наприклад, модифікованого варіанту програми QFUN на стор. 15 методичних вказівок) з урахуванням відступів у форматі команд та здатності COPASS/51 розрізняти великі та малі літери:

```
     U     BIT   28H         ; Для зручності у відповідність
     V     BIT   29H         ; значенню вхідних сигналів
     W     BIT   2AH         ; встановлюємо  біти елемента пам’яті
     X     BIT   2BH         ; з адресою 25h
     Y     BIT   2CH         ;
     Z     BIT   2DH         ;
     Q     BIT   P3.6        ;
START:MOV  P2, #00110110B    ; Імітація сигналів на входах порту P2
      MOV   25H, P2          ; Зчитування значень вхідних сигналів
      MOV   C, V             ; Занесення в біт C рівня сигналу V
      ORL   C, W             ; Логічна сума V+W в ознаці C
      ANL   C, U             ; Логічний добуток U*(V+W) в C
      MOV   F0, C            ; Збереження отриманого добутку
                             ; в біті B.0 регістра B з адресою F0h
      MOV   C, X             ; Занесення в біт C рівня сигналу X
      ANL   C, /Y            ; Логічний добуток X*У
      ORL   C, F0            ; Збереження в C суми добутків
      ORL   C, /Z            ; Збереження в C суми трьох доданків
      MOV   Q, C
```

- зберегти набраний текст програми як зміни у файлі prog0.asm виконанням опції «Сохранить» у випадному меню **Файл** вікна «Блокнот» і згорнути його натисканням кнопки «_».

3. Приєднати файл prog0.asm розробленої програми до проекту, для чого здійснити наступні дії:

- відкрити за допомогою кнопки **Project**… панелі випадних меню вікно керування файлами проекту **Project Maintenance**;

- у полі меню **File Type** поставити перемикач на той тип файлів, до якого відноситься початкова програма (у прикладі програма файлу prog0.asm написана на мові асемблера і їй відповідає тип **Assembly Source File**);
- за допомогою кнопки **Add** розгорнути вікно додавання файлів у проект **Add Project File**, після чого у полі «Имя файла» слід виділити підготовлений файл prog0.asm і додати його до проекту натисканням кнопки Ok. В результаті цих дій у полі **Project Files** вікна **Project Maintenance** будуть занесені дані щодо його розташування (за необхідності до проекту можна почергово приєднати декілька файлів із розширенням *.asm).

4. З метою отримання об’єктного модуля (файла із розширенням *.obj) приєднаного до проекту файлу prog0.asm слід виконати його трансляцію, для чого необхідно:

- увімкнути інструмент **Assembler** вибором пункту **Assembler…** випадного меню **Run** або натисканням кнопки **ASM** головної панелі інструментів, що зумовлює появу вікна **Run Assembler**;
- за допомогою кнопки **Configure Assembler** відкрити вікно конфігурації асемблера (див. рис. 4.6), вибрати необхідні опції у полі **Options** (встановленням необхідних прапорців) та параметри сторінки лістингу у полі **Listing Options** і натиснути кнопку Ok. В результаті вікно **Configure Assembler** закриється із збереженням змін;
- у вікні **Run Assembler** перенести в поле **Files** зазначений файл з поля **Available Files** шляхом його виділення і наступного натискання кнопки **Include.** За необхідності перенесення файлу із поля **Files** в поле **Available Files** слід користуватися кнопкою **Exclude (**коли необхідних файлів декілька, то третій підпункт п. 4 слід виконати необхідну кількість разів);
- у випадку, коли необхідний для трансляції файл не приєднаний до проекту (відсутній у полі **Available Files**), то його можна додати до списку **Files** файлів для транслювання натиснувши кнопку **Browse** з наступним натисканням кнопки Ok після виділенням цього файлу;
- у вікні **Run Assembler** кнопкою **Go** здійснити запуск процесу трансляції файлу (або файлів), що знаходиться у полі **Files.** В результаті трансляції буде отриманий об'єктний модуль (або модулі) із розширенням *.obj і лістинговий файл (або файли) із розширенням *.lst, які будуть розташовані у тій же папці, що і файл проекту із розширенням *.pro. При виявленні помилок у асемблерній програмі (або програмах) в основному вікні IIC з'явиться відповідне повідомлення (на-приклад, Errors 1 та Warnings 2). Більш детальний опис помилки наведений у лістинговому файлі. Для виправлення виявлених помилок у програмі її слід відкрити і скорегувати відповідним чином у текстовому редакторі «Блокнот». Після виправлення помилок програму необхідно транслювати заново.

5. Виконати опрацювання, а для декількох об'єктних модулів – їх об'єднання за допомогою укладача, в результаті чого отримати завантажувальний модуль із розширенням *.lod. Для цього слід:

- запустити інструмент **Linker** через випадне меню **Run** або кнопкою **LINC** головної панелі інструментів;

- за допомогою кнопки **Configure Linker** вікна **Run Linker** відкрити однойменне вікно конфігурації укладача та:
  - у полі **Options** поставити відмітки на всіх пунктах для створення файлу (*.map) організації адресного простору, виведення у основному вікні попереджень укладача та отримання інформації щодо опрацювання завантажувального модуля налагоджувачем;
  - у полі **Executable Format** вибрати тип майбутнього завантажувального модуля (**IEEE 695** для модуля типу *.lod та **Intel Hex-Records** для модуля типу *.ihx). Для роботи з налагоджувачем необхідний завантажувальний модуль типу **IEEE 695** із розширенням *.lod;
  - за допомогою кнопки **Configure Target** відкрити вікно конфігурації мішені (мікроконтролера);
  - встановити нульові початкові адреси пам'яті програм (ROM) і пам'яті даних зовнішніх пристроїв (XDATA);
  - у випадаючому рядку CPU вибрати тип мікроконтролера (8051);
  - у рядку Clock Speed встановити тактову частоту 7,3 Мгц і натиснути кнопку Ok для повернення у вікно **Configure Linker**;
  - у вікні **Configure Linker натиснути** кнопку Ok для збереження змін у конфігурації укладача;
- за допомогою кнопки **Include** перенести файл prog0.obj, (або декілька) з поля **Available Files** в поле **Files**. У списку **Files** головний модуль розташовують першим. Неприєднані до проекту файли можна додати до списку **Files** за допомогою кнопки **Browse**.
- за встановленої мітки у пункті **Use C Runtime Library** її слід зняти і натиснути **Go**. Якщо об'єктні модулі не містять помилок, то після виконання компоновки в основному вікні з'явиться повідомлення про успішне завершення операції і повідомлення про відсутність помилок і попереджень.

6. Перевірити працездатність програми за допомогою налагоджувача 8051 Simulator, для чого слід:

- запустити інструмент **Debugger**, через випадне меню **Run** або кнопкою **DEBUG** головної панелі інструментів. Якщо у полі Program Name не з’явиться файл завантажувального модуля (у прикладі ivanov.lod), то для цього за допомогою кнопки Browse слід вказати шлях до нього і натиснути кнопку Ok;
- увімкнути конфігуратор натисканням кнопки **Configure Debugger**, встановити в полі **Driver** драйвер симулятора (8051 Simulator) і натиснути кнопку Ok після чого у вікні **Run Debugger** слід натиснути кнопку **Go** для входження в режим налагоджування;
- відобразити необхідні інструменти налагоджувача, а саме: вікна дизасемблера та індикації регістрів спеціальних функцій – відповідним натисканням кнопок **DISASM** і **WATCH** головної панелі інструментів симулятора **51BUG Debugger V.2.30E**, вікна індикації елементів пам'яті (**ROM**, **DATA**, **XDATA**, **IDATA**), трасувальник **TRACE** та індикатор часу виконання програми **CLOCK** – перетягуванням вниз за допомогою миші верх-

ньої межі панелі завдань операційної системи та наступним подвійним натисканням лівої кнопки миші на необхідному заголовку вікна;

- перевірити роботу програми в покроковому режимі, для чого спочатку натисканням кнопки **RESET** скинути мікроконтролер у вихідний стан, після чого послідовним однократним натисканням кнопки **STEP** або клавіші F8 клавіатури по командно виконати всю програму (змінювання вмісту регістрів можна спостерігати у вікнах **REGS** і **DATA**);
- перевірити роботу програми в режимі прямого запуску на виконання, для чого послідовно: скинути мікроконтролер кнопкою **RESET**, запустити програму кнопкою **GO**, через деякий час припинити виконання програми натисканням кнопки **STOP** і переглянути результати виконання програми у вікнах інструментів налагоджувача.

7. Ознайомитись з додатковими можливостями налагоджувача, для чого:

- опанувати перегляд результатів (вміст регістрів і елементів пам’яті) і послідовності виконання команд програми в режимі анімації за допомогою наступних дій: скиду мікроконтролера натисканням кнопки **RESET**, розгортання меню **ANIMATE** з вибором інтервалу анімації, переходу в режим покрокової анімації шляхом однократного, послідовного натискання кнопок **STEPPED** і **GO**. Для зупинки програми слід натиснути кнопку **STOP**, а для поновлення режиму анімації – натиснути кнопку **GO**. Для виходу із режиму анімації повторно натискають кнопку **STEPPED**;
- опанувати процедуру змінювання вмісту необхідних елементів пам'яті (**ROM**, **DATA**, **XDATA**, **IDATA** тощо**)**, для чого подвійним натисканням лівої клавіші миші виділити одну із адрес вікна пам'яті (наприклад, **XDATA**) і замінити її на бажану (наприклад, x#0A00), після чого подібним чином виділити, а потім змінити вміст вибраного елемента (після виділення адреса та вміст забарвлюються в червоний колір);
- отримати навички змінювання формату даних у вікнах індикації вмісту елементів пам'яті, для чого послідовно розгорнути, наприклад, вікно **DATA**, змінити формат відображення вмісту регістрів з шістнадцяткової на двійкову за допомогою кнопки **MEMORY** випадного меню **Radix**, пункт **Binary** (повторити те ж для пунктів **Decimal**, **Octal**, **Hexadecimal**), за допомогою кнопки **MEMORY** (пункт **Long**) випадного меню **Data Type** змінити тип чисел на довгий (**Long**), короткий (**Short**), цілий (**Integer**), з десятковою комою звичайної (**Float**) або подвійної (**Double**) точності, байтовий тип (**Character**) чи символьний (ASCII);
- навчитися встановлювати і знімати контрольну точку, зокрема, постійну, для чого спочатку необхідно розмістити курсор в один із рядків програми у вікні дизасемблера або вікні **CODE**, а потім вибрати пункт **Toggle Permanent** випадного меню **BREAK** (в результаті у вибраному рядку з'явиться постійна контрольна точка червоного кольору). При встановленні тимчасової контрольної точки процедура подібна, однак у випадному меню **BREAK** вибирають пункт **Toggle Temporary** (після чого з'явиться контрольна точка бузкового кольору). Для видалення всіх контрольних крапок в меню **BREAK** слід вибрати **Cancel All BPs**.

- отримати навички подання чисел у різних системах числення (десятковій, шістнадцятковій, вісімковій та двійковій) за допомогою пунктів **Hexadecimal**, **Octal** і **Binary** випадного меню **EXAMINE**. Врахувати, що для переведення числа у символьний вигляд можна скористатися пунктом **Examine String** випадного меню **EXAMINE**, а для перегляду результатів перетворення – вікном **View**.

ЗАПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ

1. Що означає поняття алгебра логіки ?
2. Якими поняттями і значеннями оперує Булева алгебра?
3. Що позначається логічною одиницею і логічним нулем?
4. Який результат виконання логічних операцій X + 0 =, X + 1 =, X + X =, X · 0 =, X · X =, X · 1=?
5. У якому порядку виконують Булеві операції?
6. Які способи існують для опису функцій алгебри логіки?
7. Скільки рядків повинна містити таблиця істинності логічної функції із 3-ма вхідними змінними?
8. Для перемикальної функції Y = X3 · X2 · X1 + X3 · X2 · X1 + X3 · X2 · X1 + X3 ·X2 · X1 скласти таблицю істинності.
9. Що є метою мінімізації логічних функцій?
10. Яка послідовність підготовки завантажувального модуля?
11. Опишіть структуру текстового файлу.
12. Де може бути розташованим результат виконання операції над бітами?
13. Яка кількість базових команд бітового процесора?
14. Яка довжина команд бітового процесора?
15. Які можливості безпосереднього звертання до бітів пам’яті мікроконтролера?
16. Як звертатися до окремих бітів регістрів спеціальних функцій?
17. На які групи поділяють команди бітового процесора?
18. Які основні компоненти схеми підготовлення програм за допомогою інтегрованого інструментального середовища?
19. Які компоненти входять до складу інтегрованого інструментального середовища COMPASS/51/251?
20. Що означає вислів трансляція і для чого вона призначена?
21. Чим відрізняються емулятор і симулятор?
22. Чим обумовлюється необхідність компонування об'єктних модулів?
23. Для чого компонують файли?
24. Які основні способи пошуку логічних помилок?
25. Які існують основні групи команд COMPASS/51/251?
26. Як можна переглядати послідовність виконання команд програми?
27. Чи приводить використання кнопки Remove (видалити) вікна Project Maintenance до фізичного видалення файлу з диска?
28. Чи приводить використання кнопки Remove вікна Project Maintenance до перенесення файлу з папки проекту в іншу папку?
29. Як позначається виділений байт у вікнах налагоджувача?

30. Чи змінюється об'єктний модуль при редагуванні тексту програми у вікні DISASM налагоджувача?
31. Які умови необхідні для змінювання об'єктних модулів?
32. Чи змінюється завантажувальний модуль при редагуванні тексту програми у вікні DISASM налагоджувача?
33. Які умови необхідні для зміни завантажувального модуля?
34. Які дії необхідно виконати для установки постійної контрольної точки?
35. Які дії необхідно виконати для установки тимчасової контрольної точки?
36. Які дії необхідно виконати для скидання контрольної точки?
37. Які дії необхідно виконати для скидання всіх контрольних точок?
38. Яке призначення контрольних точок, їх типи?
39. Які є можливості змінювати вміст компонентів мікроконтролера?

Індивідуальні завдання низького рівня складності

Для отримання оцінки "*задовільно*" необхідно скласти та налагодити програму згідно із завданням (номер індивідуального завдання – за журналом групи).

У внутрішній пам'яті даних мікроконтролера, розпочинаючи з адреси adr, знаходиться масив з n однобайтових двійкових чисел вигляду $X = X_1X_0\,\text{H} = D_7D_6D_5D_4D_3D_2D_1D_0\,\text{B}$. Скласти програму перерахування елементів масиву за заданою схемою (див. таблицю) шляхом логічних дій над окремими бітами чисел. Значення бітів, не зазначених у схемі перерахування, залишаються незмінними

Таблиця

До вибору індивідуального завдання

№	Номер групи	n	adr	Схема перерахування елементів масиву
1	**2**	**3**	**4**	**5**
1	1	10	17H	$D_7 \leftarrow \overline{D_7};\ D_5 \leftarrow \overline{\overline{D_5} \vee D_3};\ D_4 \leftarrow D_4 \wedge D_3;\ D_3 \leftarrow \overline{D_2 \oplus D_1} \wedge \overline{D_5};\ D_2 \leftarrow \overline{D_7};\ D_1 \leftarrow 0$
	2	15	25H	$D_7 \leftarrow D_7 \oplus \overline{D_3};\ D_6 \leftarrow \overline{D_0};\ D_3 \leftarrow \overline{\overline{D_4} \oplus D_3};\ D_2 \leftarrow \overline{D_2};\ D_1 \leftarrow 1;\ D_0 \leftarrow D_4 \wedge \overline{D_3 \vee D_2}$
2	1	20	03H	$D_6 \leftarrow 0;\ D_5 \leftarrow \overline{D_7};\ D_4 \leftarrow D_7 \vee D_6;\ D_3 \leftarrow D_5;\ D_2 \leftarrow \overline{\overline{D_4} \wedge D_3};\ D_1 \leftarrow D_4 \vee \overline{D_2 \vee D_7}$
	2	11	10H	$D_5 \leftarrow \overline{D_5};\ D_4 \leftarrow 1;\ D_3 \leftarrow \overline{\overline{D_0} \wedge D_3};\ D_2 \leftarrow \overline{D_2 \oplus D_1} \oplus \overline{D_5};\ D_1 \leftarrow \overline{D_7};\ D_0 \leftarrow \overline{D_5} \vee \overline{D_3}$
3	1	16	15H	$D_7 \leftarrow 0;\ D_6 \leftarrow D_4 \oplus \overline{D_2 \vee D_7};\ D_5 \leftarrow \overline{D_7} \wedge D_3;\ D_4 \leftarrow \overline{D_5 \vee \overline{D_3}};\ D_2 \leftarrow D_7;\ D_0 \leftarrow \overline{D_0}$
	2	12	09H	$D_6 \leftarrow \overline{D_3 \vee \overline{D_2 \vee D_0}};\ D_5 \leftarrow 1;\ D_4 \leftarrow \overline{D_4} \wedge D_3;\ D_3 \leftarrow \overline{D_3};\ D_2 \leftarrow \overline{D_3};\ D_1 \leftarrow D_5 \oplus D_3$
4	1	17	00H	$D_6 \leftarrow 0;\ D_5 \leftarrow D_4 \oplus \overline{D_2 \vee D_0};\ D_4 \leftarrow \overline{D_4};\ D_3 \leftarrow \overline{\overline{D_4} \vee D_3};\ D_2 \leftarrow \overline{D_5 \wedge D_3};\ D_5 \leftarrow D_7$
	2	19	21H	$D_5 \leftarrow \overline{D_4 \oplus \overline{D_2 \vee D_7}};\ D_4 \leftarrow \overline{D_4};\ D_3 \leftarrow 1;\ D_2 \leftarrow \overline{D_4} \wedge D_3;\ D_1 \leftarrow D_3;\ D_0 \leftarrow \overline{D_5} \vee D_3$

Продовження таблиці

1	2	3	4	5
5	1	5	19H	$D_6 \leftarrow \overline{D_6};\ D_5 \leftarrow \overline{D_5 \oplus \overline{D}_3};\ D_4 \leftarrow \overline{D}_4 \oplus D_0;\ D_3 \leftarrow \overline{D_6 \vee \overline{D}_1} \wedge D_0;\ D_2 \leftarrow D_3;\ D_1 \leftarrow 0$
	2	7	26H	$D_7 \leftarrow D_7 \vee \overline{D}_3;\ D_6 \leftarrow \overline{D}_0;\ D_3 \leftarrow \overline{\overline{D}_4 \vee D_3};\ D_2 \leftarrow \overline{D_2};\ D_1 \leftarrow 1;\ D_0 \leftarrow D_4 \wedge \overline{D_3 \vee D_2}$
6	1	9	05H	$D_6 \leftarrow 0;\ D_5 \leftarrow \overline{D_5};\ D_4 \leftarrow \overline{D}_7 \vee D_2;\ D_3 \leftarrow D_5;\ D_2 \leftarrow \overline{D_4 \wedge \overline{D}_3};\ D_1 \leftarrow D_4 \oplus \overline{D_2 \vee D_7}$
	2	11	12H	$D_5 \leftarrow \overline{D_5};\ D_4 \leftarrow 1;\ D_3 \leftarrow D_7 \wedge \overline{D}_3;\ D_2 \leftarrow \overline{\overline{D_2 \oplus D_1} \wedge D_5};\ D_1 \leftarrow D_7;\ D_0 \leftarrow \overline{D}_5 \vee \overline{D}_3$
7	1	14	17H	$D_7 \leftarrow 0;\ D_6 \leftarrow \overline{D_4 \oplus \overline{D_2 \vee D_7}};\ D_5 \leftarrow \overline{D}_7 \oplus D_3;\ D_4 \leftarrow \overline{D_5 \vee \overline{D}_3};\ D_2 \leftarrow D_3;\ D_0 \leftarrow \overline{D_0}$
	2	16	0BH	$D_6 \leftarrow \overline{D_3 \oplus \overline{D_2 \vee D_0}};\ D_5 \leftarrow 1;\ D_4 \leftarrow \overline{D}_2 \wedge D_1;\ D_3 \leftarrow \overline{D_3};\ D_2 \leftarrow D_3;\ D_1 \leftarrow D_2 \oplus \overline{D}_3$
8	1	18	02H	$D_6 \leftarrow 0;\ D_5 \leftarrow D_3 \oplus \overline{D_2 \vee D_6};\ D_4 \leftarrow \overline{D_4};\ D_3 \leftarrow D_4 \vee D_3;\ D_2 \leftarrow \overline{\overline{D}_5 \oplus D_3};\ D_5 \leftarrow D_7$
	2	20	23H	$D_5 \leftarrow \overline{D_7 \wedge \overline{D_2 \vee D_7}};\ D_4 \leftarrow \overline{D_4};\ D_3 \leftarrow 1;\ D_2 \leftarrow \overline{\overline{D}_4 \wedge D_3};\ D_1 \leftarrow D_3;\ D_0 \leftarrow \overline{D}_5 \vee D_3$
9	1	6	03H	$D_7 \leftarrow \overline{D_7};\ D_5 \leftarrow D_7 \vee \overline{D}_3;\ D_4 \leftarrow \overline{D_4 \wedge D_3};\ D_3 \leftarrow \overline{D_4 \oplus D_0} \wedge D_4;\ D_2 \leftarrow D_7;\ D_1 \leftarrow 0$
	2	8	25H	$D_7 \leftarrow \overline{D_7 \oplus \overline{D}_3};\ D_6 \leftarrow D_0;\ D_3 \leftarrow \overline{\overline{D}_4 \oplus D_3};\ D_2 \leftarrow \overline{D_7};\ D_1 \leftarrow 1;\ D_0 \leftarrow \overline{D}_4 \oplus \overline{D_3 \vee D_2}$
10	1	10	05H	$D_6 \leftarrow 0;\ D_5 \leftarrow \overline{D_5};\ D_4 \leftarrow \overline{D}_7 \vee \overline{D}_6;\ D_3 \leftarrow D_5;\ D_2 \leftarrow D_4 \wedge D_3;\ D_1 \leftarrow D_6 \vee \overline{\overline{D}_2 \vee D_0}$
	2	12	27H	$D_5 \leftarrow \overline{D_5};\ D_4 \leftarrow 0;\ D_3 \leftarrow \overline{D_0 \wedge D_3};\ D_2 \leftarrow \overline{D_2 \wedge D_1} \oplus \overline{D}_5;\ D_1 \leftarrow \overline{D_7};\ D_0 \leftarrow \overline{D}_5 \vee \overline{D}_3$
11	1	15	07H	$D_7 \leftarrow 1;\ D_6 \leftarrow D_5 \oplus \overline{D_2 \vee D_7};\ D_5 \leftarrow \overline{D}_7 \wedge D_6;\ D_4 \leftarrow \overline{D_5 \vee \overline{D}_3};\ D_2 \leftarrow \overline{D_7};\ D_0 \leftarrow \overline{D_0}$
	2	17	21H	$D_6 \leftarrow \overline{\overline{D_7 \vee D_2} \wedge D_0};\ D_5 \leftarrow 1;\ D_4 \leftarrow \overline{D}_5 \wedge D_1;\ D_3 \leftarrow \overline{D_3};\ D_2 \leftarrow \overline{D}_3;\ D_1 \leftarrow \overline{D_5 \oplus D_3}$
12	1	20	1AH	$D_7 \leftarrow 1;\ D_6 \leftarrow D_5 \oplus \overline{D_2 \wedge D_1};\ D_5 \leftarrow \overline{D}_7 \wedge D_6;\ D_4 \leftarrow \overline{D_6 \oplus \overline{D}_3};\ D_2 \leftarrow \overline{D_7};\ D_0 \leftarrow \overline{D_0}$
	2	15	26H	$D_6 \leftarrow \overline{\overline{D_6 \vee D_2} \vee D_0};\ D_5 \leftarrow 1;\ D_4 \leftarrow \overline{\overline{D}_5 \wedge D_1};\ D_3 \leftarrow \overline{D_3};\ D_2 \leftarrow \overline{D}_3;\ D_1 \leftarrow \overline{D_5 \oplus D_3}$
13	1	10	0BH	$D_6 \leftarrow 0;\ D_5 \leftarrow \overline{D_5};\ D_4 \leftarrow \overline{\overline{D}_7 \vee D_5};\ D_3 \leftarrow D_5;\ D_2 \leftarrow D_4 \wedge D_3;\ D_1 \leftarrow D_6 \oplus \overline{\overline{D}_2 \vee D_0}$
	2	19	1CH	$D_5 \leftarrow \overline{D_5};\ D_4 \leftarrow 0;\ D_3 \leftarrow \overline{D_6 \wedge \overline{D}_2};\ D_2 \leftarrow \overline{D_4 \wedge D_1} \oplus \overline{D}_0;\ D_1 \leftarrow \overline{D_7};\ D_0 \leftarrow \overline{D}_5 \vee \overline{D}_3$

Продовження таблиці

1	2	3	4	5
14	1	14	17H	$D_7 \leftarrow \overline{D_7};\ D_5 \leftarrow \overline{D_7 \vee \overline{D}_3};\ D_4 \leftarrow \overline{D_4 \wedge D_3};\ D_3 \leftarrow \overline{D_6 \oplus D_5} \vee D_3;\ D_2 \leftarrow D_7;\ D_1 \leftarrow 0$
	2	9	0BH	$D_7 \leftarrow \overline{D_7 \vee \overline{D}_3};\ D_6 \leftarrow D_0;\ D_3 \leftarrow \overline{\overline{D}_4 \oplus D_3};\ D_2 \leftarrow \overline{D_7};\ D_1 \leftarrow 1;\ D_0 \leftarrow \overline{D_6 \oplus \overline{D_3 \vee D_2}}$
15	1	18	0DH	$D_6 \leftarrow 0;\ D_5 \leftarrow D_3 \oplus \overline{D_7 \vee D_6};\ D_4 \leftarrow \overline{D_4};\ D_3 \leftarrow D_4 \vee D_3;\ D_2 \leftarrow \overline{\overline{D}_5 \wedge D_3};\ D_5 \leftarrow D_7$
	2	12	23H	$D_5 \leftarrow \overline{D_7 \oplus \overline{D_2 \vee D_1}};\ D_4 \leftarrow \overline{D_4};\ D_3 \leftarrow 1;\ D_2 \leftarrow \overline{D_5 \wedge \overline{D}_3};\ D_1 \leftarrow D_3;\ D_0 \leftarrow \overline{D}_5 \vee D_3$
16	1	8	0EH	$D_7 \leftarrow 0;\ D_6 \leftarrow \overline{D_7 \oplus \overline{D_5 \vee D_3}};\ D_5 \leftarrow \overline{D}_7 \oplus D_3;\ D_4 \leftarrow \overline{D_5 \wedge \overline{D}_3};\ D_2 \leftarrow D_3;\ D_0 \leftarrow \overline{D_0}$
	2	17	25H	$D_6 \leftarrow \overline{D_3 \vee \overline{D_2 \oplus D_0}};\ D_5 \leftarrow 1;\ D_4 \leftarrow \overline{D}_6 \wedge D_1;\ D_3 \leftarrow \overline{D_3};\ D_2 \leftarrow D_3;\ D_1 \leftarrow D_2 \oplus \overline{D}_3$
17	1	11	0FH	$D_6 \leftarrow 0;\ D_5 \leftarrow \overline{D_5};\ D_4 \leftarrow \overline{\overline{D}_7 \vee D_6};\ D_3 \leftarrow D_5;\ D_2 \leftarrow \overline{D_4 \wedge \overline{D}_3};\ D_1 \leftarrow D_4 \oplus \overline{D_2 \vee D_7}$
	2	7	27H	$D_5 \leftarrow \overline{D_5};\ D_4 \leftarrow 1;\ D_3 \leftarrow D_7 \oplus \overline{D}_3;\ D_2 \leftarrow \overline{\overline{D_7 \oplus D_4} \wedge D_2};\ D_1 \leftarrow D_7;\ D_0 \leftarrow \overline{D}_5 \vee \overline{D}_3$
18	1	16	0EH	$D_6 \leftarrow \overline{D_6};\ D_5 \leftarrow \overline{D_5 \wedge \overline{D}_3};\ D_4 \leftarrow \overline{D}_4 \oplus D_0;\ D_3 \leftarrow \overline{D_6 \oplus \overline{D}_1} \wedge D_0;\ D_2 \leftarrow D_3;\ D_1 \leftarrow 1$
	2	10	21H	$D_7 \leftarrow \overline{D_7 \vee \overline{D}_3};\ D_6 \leftarrow D_0;\ D_3 \leftarrow \overline{\overline{D}_6 \oplus D_3};\ D_2 \leftarrow \overline{D_2};\ D_1 \leftarrow 1;\ D_0 \leftarrow D_4 \wedge \overline{D_3 \vee D_2}$

Індивідуальні завдання середнього рівня складності

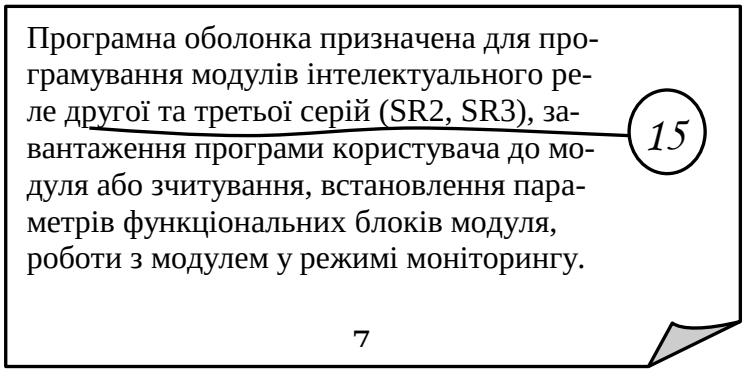
Для отримання оцінки “*добре*” треба додатково до завдання низького рівня скласти і налагодити програму обчислення молодшого байта арифметичної суми елементів вихідного масиву. Обчислений байт повинен бути записаний до елемента пам’яті, номер якого збігається з номером варіанта. Студенти, які виконують завдання з парним номером варіанта, повинні використовувати зовнішню пам’ять даних, а з непарним – внутрішню. Номер індивідуального завдання слід вибирати за журналом групи.

Індивідуальні завдання підвищеного рівня складності

Для отримання оцінки “*відмінно*” – скласти і налагодити програму сортування елементів масиву, отриманого на етапі виконання індивідуальних завдань середнього рівня складності. У випадку парного номера індивідуального завдання масив слід сортувати за збільшенням вмісту елементів, а непарного – за їх зменшенням. Окрім того, розробити програму формування масиву, який містить порядкові номери елементів скомпонованого нового масиву. Номер індивідуального завдання слід вибирати за журналом групи.

Індивідуальні завдання до самостійної роботи

Завдання полягає у опрацюванні методичних вказівок з олівцем у руках. При виконанні завдання необхідно відшукувати та підкреслювати фрагменти тексту або позначати на ілюстраціях графічні елементи, про які йдеться у запитаннях. Поруч, на березі аркуша методичних вказівок, у кружечку слід проставити номер запитання, наприклад:



Також у графі «Посилання» треба зазначити номер сторінки, на якій знайдено відповідь на запитання.

Увага! Виконання наведених завдань дозволяється лише у власному примірнику методичних вказівок. Забороняється проставляти будь-які позначки або виконувати креслення у методичних вказівках бібліотечного фонду.

Індивідуальні варіанти завдань до самостійної роботи

№ зап.	№ вар.	Завдання або запитання	Посилання
	1/1	Знайдіть та підкресліть у тексті методичних вказівок визначення булевої алгебри.	
	2/2	Знайдіть та позначте у тексті методичних вказівок таблицю істинності операції логічного додавання.	
	3/3	Знайдіть та позначте у тексті методичних вказівок електричну схему, що реалізує операцію логічного додавання.	
	4/4	Знайдіть та позначте у тексті методичних вказівок умовне позначення логічного елемента “АБО”.	
	5/5	Знайдіть та позначте у тексті методичних вказівок таблицю істинності операції логічного множення.	
	6/6	Знайдіть та позначте у тексті методичних вказівок електричну схему, що реалізує операцію логічного множення.	
	7/7	Знайдіть та позначте у тексті методичних вказівок умовне позначення логічного елемента “І”.	
	8/8	Знайдіть та підкресліть у тексті методичних вказівок визначення операції логічного заперечення.	
	9/9	Знайдіть та підкресліть у тексті методичних вказівок визначення операції інверсії.	
	10/10	Позначте формули, які утілюють комутативний закон алгебри логіки.	
	11/11	Позначте формули, які утілюють закон перестановки алгебри логіки.	
	12/12	Позначте формули, які утілюють асоціативний закон алгебри логіки.	
	13/13	Позначте формули, які утілюють закон сполучення алгебри логіки.	
	14/14	Позначте формули, які утілюють дистрибутивний закон алгебри логіки.	
	15/15	Позначте формули, які утілюють закон розподілу алгебри логіки.	
	1/16	Позначте формули, які утілюють закон інверсії алгебри логіки.	
	2/17	Позначте формули, які утілюють правило Моргана.	
	3/18	Позначте формули, які утілюють закон подвійної інверсії алгебри логіки.	
	4/19	Позначте формули, які утілюють закон подвійного заперечення алгебри логіки.	
	5/20	Позначте формули, які утілюють закон повторення алгебри логіки.	
	6/1	Позначте формули, які утілюють закон індемпотентності.	
	7/2	Позначте формули, які утілюють закон суперечності.	
	8/3	Позначте формули, які утілюють закон виключення третього.	
	9/4	Позначте формули, які утілюють закон склеювання.	
	10/5	Позначте формули, які утілюють закон поглинання.	
	11/6	Позначте формули, які утілюють закон незмінності.	
	12/7	Позначте формули, які утілюють закон універсальної і нульової множини.	
	13/8	Знайдіть та позначте фрагмент тексту у якому йдеться про мету мінімізації булевих функцій.	
	14/9	Знайдіть та позначте фрагмент тексту, який показує на чому базуються табличні методи мінімізації логічних виразів.	
	15/10	Знайдіть та позначте фрагмент тексту, який показує на чому базуються аналітичні методи мінімізації логічних виразів.	
	1/11	Позначте на рис. 2.1 функції збудження бістабільної ланки.	
	2/12	Позначте на рис. 2.2 комірки, у яких функція може приймати значення або 0 або 1.	
	3/13	Позначте на рис. 2.3 вхід встановлення тригера.	
	4/14	Позначте на рис. 2.3 вхід скидання тригера.	
	5/15	Позначте на рис. 2.3 тактовий вхід тригера.	
	6/16	Позначте фрагмент тексту у якому йдеться про кількість команд бітового процесора мікроконтролера MCS-51.	
	7/17	Позначте фрагмент тексту у якому надане визначення бітового процесора мікроконтролера MCS-51.	
	8/18	Позначте на рис. 2.4 біт D7 елемента ОЗП з адресою 2Ch. Запишіть сюди пряму адресу цього біта: .	

	9/19	Позначте на рис. 2.4 біт D5 елемента ОЗП з адресою 27h. Запишіть сюди пряму адресу цього біта: .	
	10/20	Позначте на рис. 2.4 біт D3 елемента ОЗП з адресою 25h. Запишіть сюди пряму адресу цього біта: .	
	11/1	Позначте на рис. 2.4 біт D1 елемента ОЗП з адресою 23h. Запишіть сюди пряму адресу цього біта: .	
	12/2	Позначте на рис. 2.4 біт D6 елемента ОЗП з адресою 21h. Запишіть сюди пряму адресу цього біта: .	
	13/3	Позначте на рис. 2.4 біт D4 елемента ОЗП з адресою 2Ah. Запишіть сюди пряму адресу цього біта: .	
	14/4	Позначте на рис. 2.4 біт D2 елемента ОЗП з адресою 2Bh. Запишіть сюди пряму адресу цього біта: .	
	15/5	Позначте на рис. 2.4 біт D0 елемента ОЗП з адресою 2Dh. Запишіть сюди пряму адресу цього біта: .	
	1/6	Позначте на рис. 2.4 біт D3 елемента ОЗП з адресою 2Eh. Запишіть сюди пряму адресу цього біта: .	
	2/7	Позначте на рис. 2.4 біт D5 елемента ОЗП з адресою 29h. Запишіть сюди пряму адресу цього біта: .	
	3/8	Позначте на рис. 2.4 біт D7 елемента ОЗП з адресою 21h. Запишіть сюди пряму адресу цього біта: .	
	4/9	Позначте на рис. 2.5 біт D0 акумулятора. Запишіть сюди пряму адресу цього біта: .	
	5/10	Позначте на рис. 2.5 біт D2 регістра-розширювача акумулятора. Запишіть сюди пряму адресу цього біта: .	
	6/11	Позначте на рис. 2.5 біт D4 регістра SFR нульового порта. Запишіть сюди пряму адресу цього біта: .	
	7/12	Позначте на рис. 2.5 біт D6 регістра SFR першого порта. Запишіть сюди пряму адресу цього біта: .	
	8/13	Позначте на рис. 2.5 біт D1 регістра SFR другого порта. Запишіть сюди пряму адресу цього біта: .	
	9/14	Позначте на рис. 2.5 біт D3 регістра SFR третього порта. Запишіть сюди пряму адресу цього біта: .	
	10/15	Позначте на рис. 2.5 біт D5 регістра слова стану програми. Запишіть сюди пряму адресу цього біта: .	
	11/16	Позначте на рис. 2.5 біт, що містить прапорець перенесення. Запишіть сюди пряму адресу цього біта: .	
	12/17	Позначте на рис. 2.5 біт, що містить прапорець допоміжного перенесення. Запишіть сюди пряму адресу цього біта: .	
	13/18	Позначте на рис. 2.5 біт, що містить прапорець переповнення. Запишіть сюди пряму адресу цього біта: .	
	14/19	Позначте на рис. 2.5 біт, що містить прапорець паритету. Запишіть сюди пряму адресу цього біта: .	
	15/20	Позначте на рис. 2.5 біт, що містить прапорець користувача. Запишіть сюди пряму адресу цього біта: .	
	1/1	Позначте на рис. 2.5 біт, що містить нульовий біт селектора банків регістрів. Запишіть сюди пряму адресу цього біта: .	
	2/2	Позначте на рис. 2.5 біт, що містить старший біт селектора банків регістрів. Запишіть сюди пряму адресу цього біта: .	
	3/3	Позначте на рис. 2.5 зарезервований біт слова стану програми. Запишіть сюди пряму адресу цього біта: .	
	4/4	Позначте на рис. 2.6 входи, що реалізують функцію заперечення.	
	5/5	Позначте на рис. 2.6 входи, що реалізують функцію інверсії.	
	6/6	Позначте на рис. 2.6 елементи, що виконують операцію диз'юнкції.	
	7/7	Позначте на рис. 2.6 елементи, що виконують операцію кон'юнкції.	
	8/8	У програмі QFUN позначте будь-яку команду, що виконує інверсію одного з операндів.	
	9/9	У програмі QFUN позначте будь-яку команду, що використовує біт прапорця перенесення.	

	10/10	У програмі QFUN позначте будь-яку команду, що відноситься до групи команд пересилки даних.	
	11/11	У програмі QFUN позначте будь-яку команду, що виконує операцію диз'юнкції.	
	12/12	У програмі QFUN позначте будь-яку команду, що виконує операцію кон'юнкції.	
	13/13	Позначте фрагмент тексту, у якому йдеться про обмеження реалізації комбінаційної схеми із значною кількістю входів/виходів за допомогою однокристального мікроконтролера.	
	14/14	Позначте у тексті методичних вказівок типові операції при розробці програми за допомогою IIC.	
	15/15	Запишіть сюди назву інструмента IIC або операційної системи, який використовується для введення тексту програми: .	
	1/16	Запишіть сюди назву інструмента IIC, який використовується для трансляції програми з файлу, що написаний мовою високого рівня: .	
	2/17	Запишіть сюди назву інструмента IIC, який використовується для трансляції програми з файлу, що написаний мовою низького рівня: .	
	3/18	Запишіть сюди назву інструмента IIC, який використовується для компонування об'єктних модулей: .	
	4/19	Запишіть сюди назву інструмента IIC, який використовується для створення завантажувального файлу: .	
	5/20	Запишіть сюди назву інструмента IIC, який використовується для пошуку логічних помилок у програмі користувача: .	
	6/1	Позначте фрагмент тексту, у якому надане визначення текстового файлу.	
	7/2	Позначте фрагмент тексту, у якому йдеться про можливі розширення імен файлів вихідних програм.	
	8/3	Позначте фрагмент тексту, у якому описаний порядок дій для перегляду файлу роздруку.	
	9/4	Позначте фрагмент тексту, у якому йдеться про формування імені завантажувального модуля.	
	10/5	Позначте фрагмент тексту, у якому йдеться про переміщуваність завантажувального модуля.	
	11/6	Позначте фрагмент тексту, у якому дається визначення константи зміщення.	
	12/7	Позначте формулу для розрахунку абсолютних адреси позначок у завантажувальному модулі.	
	13/8	Позначте правила, що покладені в основу принципу переміщуваності модулів.	
	14/9	Позначте на рис. 3.3 вікно у якому задається початкова адреса ПЗП.	
	15/10	Позначте на рис. 3.3 вікно у якому задається початкова адреса ОЗП.	
	1/11	Позначте на рис. 3.3 вікно у якому задається кінцева адреса ПЗП.	
	2/12	Позначте на рис. 3.3 вікно у якому задається кінцева адреса ОЗП.	
	3/13	Позначте на рис. 3.3 вікно у якому задається початкова адреса зовнішньої пам'яті даних.	
	4/14	Позначте на рис. 3.3 вікно у якому задається кінцева адреса зовнішньої пам'яті даних.	
	5/15	Позначте на рис. 3.3 вікно у якому можна обрати тип мікроконтролера.	
	6/16	Позначте на рис. 3.3 вікно у якому можна задати тактову частоту мікроконтролера.	
	7/17	Позначте у тексті методичних вказівок способи виявлення та локалізації логічних помилок у програмі користувача.	
	8/18	Знайдіть та позначте у тексті методичних вказівок перелік інструментів інтегрального інструментального середовища COMPASS 51 IDE.	
	9/19	Позначте у тексті методичних вказівок визначення симулятора.	
	10/20	На рис. 4.1 позначте кнопку виклику вікна менеджера (диспетчера) файлів.	
	11/1	На рис. 4.1 позначте кнопку виклику вікна вибору конфігурації середовища (не плутати з кнопкою вибору конфігурації інструментів).	
	12/2	На рис. 4.1 позначте кнопку виклику вікна підключення файлів та бібліотек до проекту.	
	13/3	На рис. 4.1 позначте кнопку вставки поточної дати та часу у головне вікно IIC.	
	14/4	На рис. 4.1 позначте кнопку очищення головного вікна IIC.	
	15/5	На рис. 4.1 позначте кнопку виклику вікна вибору конфігурації інструментів IIC (не плутати з вікном вибору конфігурації мішені).	
	1/6	На рис. 4.1 позначте кнопку виклику вікна запуску компілятора.	
	2/7	На рис. 4.1 позначте кнопку виклику вікна запуску асемблера.	
	3/8	На рис. 4.1 позначте кнопку виклику вікна запуску укладача.	
	4/9	На рис. 4.1 позначте кнопку виклику вікна запуску налагоджувача.	

	5/10	На рис. 4.1 позначте кнопку виклику вікна запуску текстового редактора.	
	6/11	На рис. 4.1 позначте кнопку виклику вікна стану задач.	
	7/12	На рис. 4.1 позначте кнопку виклику вікна вибору конфігурації мішені (не плутати з вікном вибору конфігурації інструментів IIC).	
	8/13	У переліку команд меню головного вікна IIC позначте команду перегляду файлу за допомогою текстового редактора.	
	9/14	У переліку команд меню головного вікна IIC позначте команду запису проекту із заданим ім'ям.	
	10/15	У переліку команд меню головного вікна IIC позначте команду роздруку вмісту головного вікна IIC.	
	11/16	У переліку команд меню головного вікна IIC позначте команду очищення головного вікна IIC.	
	12/17	У переліку команд меню головного вікна IIC позначте команду вставки поточної дати та часу у головне вікно IIC.	
	13/18	У переліку команд меню головного вікна IIC позначте команду виклику вікна складу проекту.	
	14/19	На рис. 4.2 позначте список файлів проекту.	
	15/20	На рис. 4.2 позначте радіокнопку перемикача типів файлів проекту.	
	1/1	На рис. 4.2 позначте кнопку додавання файла до поточного проекту.	
	2/2	На рис. 4.2 позначте кнопку видалення файла із списка файлів проекту.	
	3/3	На рис. 4.2 позначте кнопку додавання бібліотеки до поточного проекту.	
	4/4	На рис. 4.2 позначте кнопку видалення бібліотеки із списка бібліотек проекту.	
	5/5	У переліку команд меню головного вікна IIC позначте команду виклику вікна вибору конфігурації мішені.	
	6/6	У переліку команд меню головного вікна IIC позначте команду виклику вікна вибору конфігурації середовища.	
	7/7	У переліку команд меню головного вікна IIC позначте команду виклику вікна стану задач.	
	8/8	На рис. 4.3 позначте список доступних файлів проекту.	
	9/9	На рис. 4.3 позначте список файлів, що обробляються компілятором.	
	10/10	На рис. 4.3 позначте кнопку включення файлу до складу компілюємих.	
	11/11	На рис. 4.3 позначте кнопку виключення файлу із складу компілюємих.	
	12/12	На рис. 4.5 позначте текстовий рядок, до якого вводиться розширення імені файла.	
	13/13	На рис. 4.5 позначте текстовий рядок, у якому відображається програма, що співставлена заданому розширенню імені файла.	
	14/14	На рис. 4.6 позначте прапорець, який слід встановити для генерування об'єктного модуля.	
	15/15	На рис. 4.6 позначте прапорець, який слід встановити для генерування лістингового файлу.	
	1/16	На рис. 4.6 позначте прапорець, який слід встановити для включення макросів до лістингу.	
	2/17	На рис. 4.6 позначте прапорець, який слід встановити для виводу до головного вікна IIC лише повідомлень про помилки та попереджень.	
	3/18	На рис. 4.6 позначте прапорець, який слід встановити, якщо необхідно розрізняти регістр літер у іменах змінних.	
	4/19	На рис. 4.6 позначте прапорець, який слід встановити для увімкнення режиму Intel-сумісності при асемблюванні.	
	5/20	На рис. 4.6 позначте текстовий рядок, до якого слід вводити бажану кількість рядків на сторінку у файлі роздруку.	
	6/1	На рис. 4.6 позначте текстовий рядок, до якого слід вводити бажану кількість символів на рядок у файлі роздруку.	
	7/2	У тексті методичних вказівок позначте типову структуру одного рядка тексту асемблерної програми.	
	8/3	На рис. 4.7 позначте комбінований список, у якому можна вибрати тип файлу для компонування.	
	9/4	На рис. 4.7 позначте текстовий рядок, до якого слід вводити бажане ім'я завантажувального модуля, який буде створений укладачем.	
	10/5	На рис. 4.7 позначте кнопку виклику вікна вибору конфігурації укладача.	
	11/6	На рис. 4.8 позначте комбінований список, у якому можна вибрати формат завантажувального модуля, що буде створений укладачем.	
	12/7	На рис. 4.8 позначте кнопку виклику вікна вибору конфігурації мішені.	

	13/8	На рис. 4.10 позначте прапорець, встановлення якого запускає укладач при виконанні проекту.	
	14/9	На рис. 4.10 позначте назву інструмента IIC, який виконує трансляцію програми користувача, що написана мовою високого рівня.	
	15/10	На рис. 4.10 позначте назву інструмента IIC, який виконує трансляцію програми користувача, що написана мовою низького рівня.	
	1/11	На рис. 4.10 позначте назву інструмента IIC, який виконує об'єднання окремих об'єктних модулів.	
	2/12	На рис. 4.11 позначте комбінований список для вибору драйвера налагоджувача.	
	3/13	На рис. 4.11 позначте прапорець, встановлення якого дозволяє зберігати вигляд та розташування вікон налагоджувача.	
	4/14	На рис. 4.12 позначте вікно, у якому відображається вміст пам'яті програм мікропроцесорної системи.	
	5/15	На рис. 4.12 позначте вікно, у якому відображається вміст зовнішньої пам'яті даних мікропроцесорної системи.	
	6/16	На рис. 4.12 позначте вікно, у якому відображається вміст внутрішньої пам'яті даних мікроконтролера.	
	7/17	На рис. 4.12 позначте вікно, у якому відображається вміст регістрів спеціальних функцій мікроконтролера.	
	8/18	На рис. 4.12 позначте вікно дизасемблера.	
	9/19	На рис. 4.12 позначте вікно, у якому виводяться повдомлення налагоджувача.	
	10/20	На рис. 4.12 позначте вікно – командний рядок налагоджувача.	
	11/1	На рис. 4.12 позначте вікно – індикатор кількості виконаних машинних циклів та поточного часу виконання програми.	
	12/2	На рис. 4.12 позначте вікно для перегляду вмісту обраних елементів або регістрів мікроконтролера.	
	13/3	На рис. 4.12 позначте кнопку, що дозволяє виконати одну поточну інструкцію програми користувача.	
	14/4	На рис. 4.12 позначте кнопку, що дозволяє виконати команди тіла циклу або підпрограму без детального перегляду.	
	15/5	На рис. 4.12 позначте кнопку, що імітує апаратне перезавантаження мікроконтролера.	
	1/6	На рис. 4.12 позначте кнопку, що дозволяє переводити числа з одної системи числення до іншої.	
	2/7	На рис. 4.12 позначте кнопку, що дозволяє змінювати формат подання даних у вікнах вмісту пам'яті.	
	3/8	На рис. 4.12 позначте кнопку, що дозволяє оперувати з контрольними точками.	
	4/9	На рис. 4.12 позначте кнопку, що дозволяє увімкнути або вимкнути режим часової анімації.	
	5/10	На рис. 4.12 позначте кнопку, що дозволяє увімкнути або вимкнути режим покрокової анімації.	
	6/11	На рис. 4.12 позначте кнопку, яка надає швидку підказку щодо "гарячих" клавіш налагоджувача.	
	7/12	Позначте фрагмент тексту який відображає порядок дій для встановлення контрольних точок у середовищі COMPASS/51 IDE.	
	8/13	Позначте фрагмент тексту який відображає порядок дій для занесення регістрів чи елементів пам'яті до вікна WATCH у середовищі COMPASS/51 IDE.	
	9/14	У переліку команд командного рядка налагоджувача позначте команду запуску завантаженного модуля на виконання.	
	10/15	У переліку команд командного рядка налагоджувача позначте команду виконання однієї поточної підпрограми, циклу або однієї команди, що не належить до підпрограми або циклу.	
	11/16	У переліку команд командного рядка налагоджувача позначте команду виконання однієї асемблерної інструкції програми користувача.	
	12/17	У переліку команд командного рядка налагоджувача позначте команду, що дозволяє імітувати апаратний перезапуск мікроконтролера.	
	13/18	У переліку команд командного рядка налагоджувача позначте команду виходу із оболонки налагоджувача.	
	14/19	Позначте фрагмент тексту, у якому йдеться про призначення вікна статусу задач COMPASS/51 IDE.	
	15/20	Позначте фрагмент тексту, у якому наданий перелік кнопок і елементів вікна статусу задач COMPASS/51 IDE.	

Упорядники:
Кириченко Віталій Іванович
Яланський Олексій Анатолійович
Алпаєв Володимир Георгієвич

Методичні вказівки до виконання лабораторної роботи МПП-3 «Бітовий процесор мікроконтролера 8051АН сімейства MCS-51. Програмування в інтегрованому інструментальному середовищі IDE COMPASS/51/251», індивідуальних завдань та самостійної роботи з професійно-орієнтованої дисципліни «Мікропроцесорні пристрої»

для студентів спеціальності 7.092203 «Електромеханічні системи автоматизації та електропривод» напряму «Електромеханіка»

Редакційно-видавничий комплекс
Друкується у редакційній обробці упорядників

Підписано до друку . Формат 30×42/4.
Папір офсет. Ризографія. Умовн. друк. арк. .
Обліково-видавн. арк. . Тираж 100 прим. Зам. №

НГУ
49027, м. Дніпропетровськ-27, просп. К. Маркса, 19.