Міністерство освіти і науки, молоді та спорту України
Державний вищий навчальний заклад
„Національний гірничий університет"

Кафедра систем електропостачання

Методичні вказівки

до виконання лабораторної роботи МП-7

„Програмування мікропроцесора КР580ВМ80 з використанням емулятора CPU580"

для студентів, які навчаються за напрямами підготовки:
6.050701 „Електротехніка та електроенергетика ",
6.050702 „Електромеханіка "

Дніпропетровськ
2012

Методичні вказівки до виконання лабораторної роботи 7-МП "Програмування мікропроцесора КР580ВМ80 з використанням емулятора CPU580" для студентів, які навчаються за напрямами підготовки: 6.050701 "Електротехніка та електроенергетика", 6.050702 "Електромеханіка" / Укл.: Г.М.Бажін, А.В.Рухлов, С.В.Дибрін, Д.О.Кошовий. – Дніпропетровськ: Державний ВНЗ "НГУ", кафедра систем електропостачання, 2012. - 17 с.

1. Мета роботи

Вивчити: систему команд мікропроцесора КР580ВМ80, методику складання простих програм, їх асемблювання, запуск в роботу й отримання результату.

У процесі виконання роботи студент повинен:

– вивчити архітектуру мікропроцесора КР580ВМ80, систему його команд;

– уміти складати прості лінійні програми для мікропроцесора КР580ВМ80, асемблювати їх і запускати в роботу за допомогою емулятора CPU580.

2. Теоретичні положення роботи

2.1. Архітектура мікропроцесора

Мікропроцесор має складну внутрішню структуру, проте, з точки зору програміста, він складається з декількох регістрів. До їх числа (рис. 1) входять: акумулятор, шість 8-розрядних регістрів загального призначення, регістр ознак і два 16-розрядні регістри – вказівник стека (SP) та лічильник команд (PC).

Акумулятор (A) призначено для зберігання одного з операндів перед виконанням арифметико-логічних операцій, а також результату, отриманого після їх виконання. 8-розрядні регістри загального призначення прийнято позначати латинськими буквами B, C, D, E, H, і L. Вони призначаються для тимчасового зберігання проміжних даних. Регістри програмно доступні. При виконанні деяких команд регістри B і C, D і E, H і L об'єднуються в регістрові пари BC, DE і HL. Це дозволяє використовувати їх для зберігання адреси або 16-розрядних операндів.

Регістр ознак (F) призначений для зберігання ознак, що характеризують результат виконання арифметичних і логічних команд. Тут зберігаються такі ознаки:

C – ознака перенесення зі старшого 8-го розряду в 9-й при виконанні арифметичних операцій (переповнювання акумулятора);

P – ознака парності (парної кількості одиниць) в отриманому результаті;

AC – ознака наявності допоміжного перенесення (з 3-го розряду в 4-й);

Z – ознака наявності нульового результату;

S – ознака знаку (результат має позитивний або негативний знак).

Вказівник стека (SP) – це програмно доступний регістр, який вказує на верхівку стека. Стек – це область пам'яті, яка виділяється для тимчасового зберігання вмісту регістрових пар або слова стану процесора.

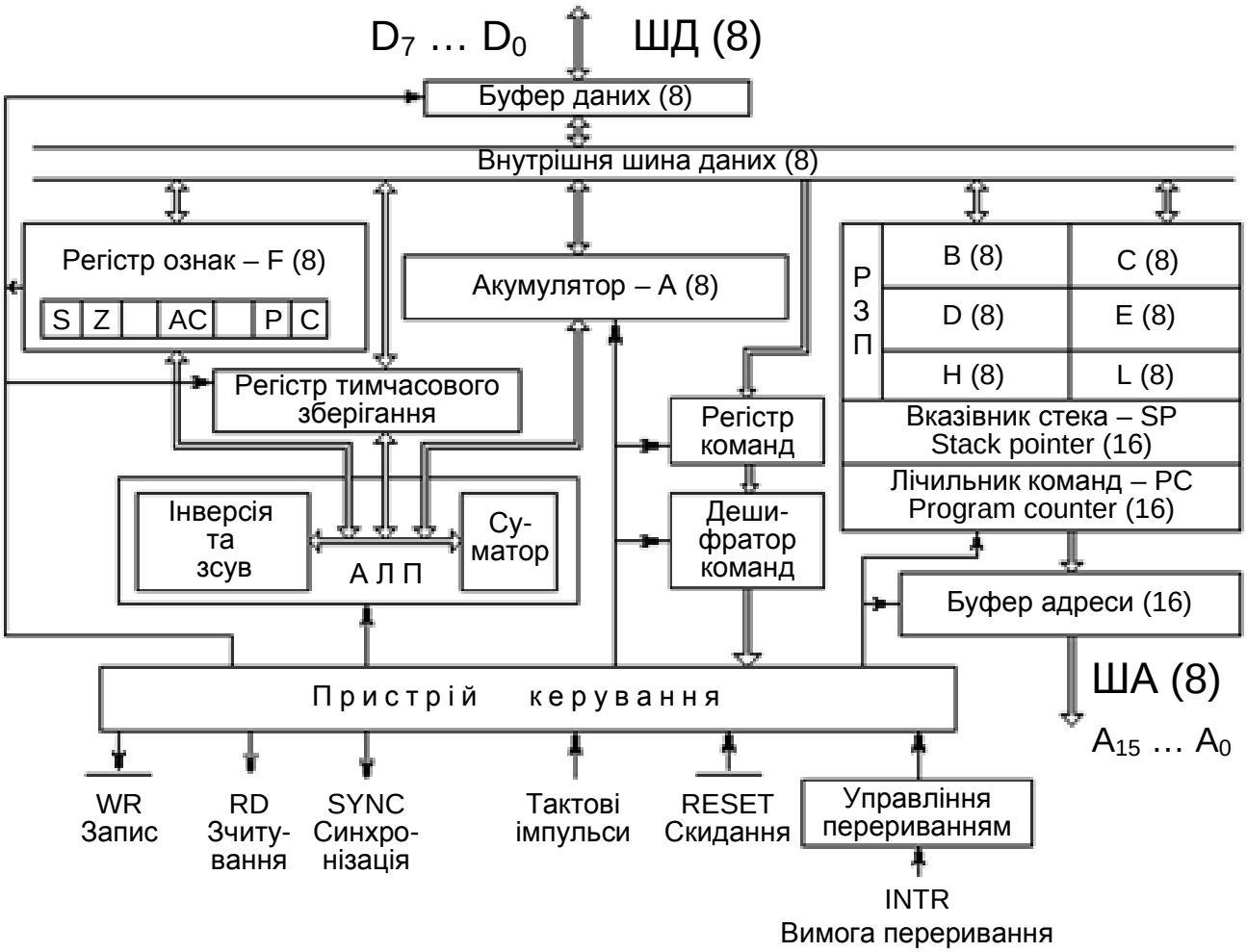

Рис. 1. Архітектура мікропроцесора серії КР580ВМ80

Лічильник команд (PC) – програмно доступний регістр, в якому зберігається адреса команди, що виконуватиметься в наступному кроці.

Запустивши програму CPU580 і вибравши в меню основного вікна позицію СТРУКТУРНАЯ СХЕМА, отримаємо на екрані структурну схему мікропроцесора, аналогічну тій, що показана на рис. 1. Завдяки їй можна переглядати процес і результати виконання програми. Повернувшись через клавішу ВЫХОД до основного вікна емулятора та вибравши в меню позицію СИСТЕМА КОМАНД, можна викликати на екран таблицю команд. Принципи її використання будуть розглянуті пізніше. Повернення до основного вікна емулятора – через клавішу ОК.

2.2. Система команд мікропроцесора

Усі команди мікропроцесора наведено в табл. 1. Суть кожної команди вказано після крапки з комою. Умовні позначення розшифровано в розділі "умовні позначення" у нижній частині таблиці. Коди команд наведено в табл. 2. Методику користування таблицею розглянемо на прикладі.

Таблиця 1. Система команд мікропроцесора КР580ВМ80

Однобайтові пересилання

MOV R1, R;	R → R1
MVI R, D8;	D8 → R
STA adr;	A → M(adr)
LDA adr;	M(adr) → A
STAX YZ+;	A → M(YZ+)
LDAX YZ+;	M(YZ+) → A

Двобайтові пересилання

SPHL;	HL → SP
LXI YZ, D16;	D16 → YZ
SHLD adr;	L → M(adr), H → M(adr+1)
LHLD adr;	M(adr) → L, M(adr+1) → H
PUSH YZ++;	Y →M(SP-1); Z →M(SP-2); SP-2→SP
PUSH PSW;	A → M(SP-1); F →M(SP-2); SP-2 → SP
POP YZ++;	M(SP)→Z; M(SP+1)→Y; SP+2→SP
POP PSW;	M(SP)→F; M(SP+1)→A; SP+2→SP

Команди введення і виводу

IN N	(N) → A
OUT N;	A → (N)

Обмін байтами

XCHG;	HL←→DE
XTHL;	H←→M(SP+1), L←→M(SP)

Арифметичні і логічні операції з одним операндом

CMC";	$C \rightarrow \overline{C}$
STC";	1 → C
(C - ознака перенесення)	
CMA;	$A \rightarrow \overline{A}$
DAA';	Десяткова корекція
INR"' R;	R+1 → R
DCR"' R ;	R - 1 → R
INX YZ;	YZ+1 → YZ
DCX YZ;	YZ - 1 → YZ

Арифметичні і логічні операції з двома операндами

ADD' R;	A+ R → A	ADI' D8;	A+ D8 → A
ADC' R;	A+ (R+C) → A	ACI' D8;	A+ (D8+C) → A
SUB' R;	A - R → A	SUI' D8;	A - D8 → A
SBB' R;	A - (R+C) → A	SBI' D8;	A - (D8+C) → A
ANA' R;	A∪R → A;	ANI' D8	A∪D8 → A
ORA' R;	A∩R → A	ORI' D8;	A∩D8 → A
XRA' R;	A⊕(R → A	XRI' D8;	A⊕D8 → A

(тут C – ознака перенесення в регістрі F)

Установка ознак регістра F відповідно до

CPI' D8; A - D8

або

CMP' R A - R

16-бітові операції

DAD" YZ; HL+YZ → HL

Команди зсуву вмісту акумулятора

RLC"; Зсув ліворуч
RAL"; Зсув ліворуч через біт ознаки C
RRC"; Зсув праворуч
RAR"; Зсув праворуч через біт ознаки C

Команди передачі управління

PCHL;	HL → PC
JMP adr;	adr → PC
J-CON adr;	adr → PC

Спеціальні команди

EI;	Дозвіл переривання
DI;	Заборона переривання
HLT;	Зупинка
NOP;	Холоста операція

Формат регістра F

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	0	AC	0	P	1	C

Команди виклику і повернення з п/п

CALL adr;
C - CON adr; } PC→M(SP - 1) M(SP - 2); adr →PC

RST X;
де X = 0, 1,…, 7 } { PC → M(SP - 1) M(SP - 2)
adr → PC

Для кожного X адреса (adr), відповідно, дорівнює 0H;8H;10H;18H;20H;28H;30H;38H;

RET;
R - CON; } M(SP) M(SP+1)→PC; SP+2→SP

Умовні позначення:

' – команда чинить дію на всі ознаки регістра F;
" – команда чинить дію тільки на ознаку C (перенесення до 8-го розряду або переповнювання);
"' – команда чинить дію на всі ознаки, крім ознаки C;
R, R1 – вміст регістрів A, B, C, D, E, H, L або комірки пам'яті M(HL);
YZ – вміст регістрової пари BC, DE, HL або покажчика стека SP (YZ в мнемоніці замінюється на B, D, H або SP);
YZ+ – вміст регістрової пари BC або DE (YZ+ в мнемоніці замінюється на B або D);
YZ++ – вміст регістрової пари BC, DE або HL (YZ++ в мнемоніці замінюється на B, D або H);
SP – вміст покажчика стека перед виконанням команди;
D8 – 8-розрядний операнд (вказується в другому байті команди);
(N) – вміст порту вводу або виводу з номером N = 0; 1;...; 255;
D16 – 16-розрядний операнд (вказується в другому і третьому байтах команди);
adr – 16-розрядна адреса;
M(…) – вміст комірки пам'яті (адреса вказується в дужках);
-CON – частина мнемоніки команди, що визначає умову передачі управління, виклику і повернення з підпрограми (-CON в мнемоніці замінюється на NZ, Z, NC, C, PO, PE, P, M).

Таблиця 2. Коди команд мікропроцесора КР580ВМ80

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LXI B,&	STAX B	INX B	INR B	DCR B	MVI B,#	RLC	-	DAD B	LDAX B	DCX B	INR C	DCR C	MVI C,#	RRC
1	-	LXI D,&	STAX D	INX D	INR D	DCR D	MVI D,#	RAL	-	DAD D	LDAX D	DCX D	INR E	DCR E	MVI E,#	RAR
2	-	LXI H,&	SHLD *	INX H	INR H	DCR H	MVI H,#	DAA	-	DAD H	LHLD *	DCX H	INR L	DCR L	MVI L,#	CMA
3	-	LXI SP&	STA *	INX SP	INR M	DCR M	MVI M,#	STC	-	DAD SP	LDA *	DCX SP	INR A	DCR A	MVI A,#	CMC
4	MOV B, B	MOV B, C	MOV B, D	MOV B, E	MOV B, H	MOV B, L	MOV B, M	MOV B, A	MOV C, B	MOV C, C	MOV C, D	MOV C, E	MOV C, H	MOV C, L	MOV C, M	MOV C, A
5	MOV D, B	MOV D, C	MOV D, D	MOV D, E	MOV D, H	MOV D, L	MOV D, M	MOV D, A	MOV E, B	MOV E, C	MOV E, D	MOV E, E	MOV E, H	MOV E, L	MOV E, M	MOV E, A
6	MOV H, B	MOV H, C	MOV H, D	MOV H, E	MOV H, H	MOV H, L	MOV H, M	MOV H, A	MOV L, B	MOV L, C	MOV L, D	MOV L, E	MOV L, H	MOV L, L	MOV L, M	MOV L, A
7	MOV M, B	MOV M, C	MOV M, D	MOV M, E	MOV M, H	MOV M, L	HLT	MOV M, A	MOV A, B	MOV A, C	MOV A, D	MOV A, E	MOV A, H	MOV A, L	MOV A, M	MOV A, A
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC M	ADC A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A
B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A
C	RNZ	POP B	JNZ *	JMP *	CNZ *	PUSH B	ADI #	RST 0	RZ	RET	JZ *	-	CZ *	CAL L	ACI #	RST 1
D	RNC	POP D	JNC *	OUT N	CNC *	PUSH D	SUI #	RST 2	RS	-	JC *	IN N	CC *	* -	SBI *	RST 3
E	RPO	POP H	JPO *	XTH L	CPO *	PUSH H	ANI #	RST 4	RPE	PCHL	JPE *	XCHG	CPE *	-	XRI #	RST 5
F	RP	POP PSW	JP *	DI	CP *	PUSH PSW	ORI #	RST 6	RM	SPHL	JM *	EI	CM *	-	CPI #	RST 7
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Примітки:

N - номер порту введення-виводу;

& – двобайтовий операнд D16;

* – двобайтовий операнд adr;

– однобайтовий операнд D8;

PSW - вміст акумулятора і регістра ознак (слово стану процесора).

ПРИКЛАД: код операції – C3; операція – JMP adr.

Приклад 1. Користуючись табл. 1, визначити призначення команди STAX D. У розділі "Однобайтові пересилання" знаходимо рядок, що містить мнемоніку STAX. У цьому рядку записано **STAX YZ^+; A → M(YZ^+)**. В умовних позначеннях читаємо: YZ^+ – вміст регістрової пари BC або DE, причому в мнемоніці BC або DE замінюється на B або D. Запис після крапки з комою означає, що вміст акумулятора переміщується (або ще говорять завантажується) до комірки пам'яті (згідно з умовними позначеннями літера M означає пам'ять – від слова memory). У дужках вказується адреса комірки пам'яті. Оскільки в нашому прикладі в дужках вказане YZ^+, тобто бере участь регістрова пара BC або DE, тоді адреса знаходитиметься в одній з цих пар. Оскільки в мнемоніці початкової команди використовується D, це вказує на регістрову пару DE.

Таким чином, команда STAX D означає пересилку вмісту акумулятора до комірки пам'яті. Адреса цієї комірки міститься в регістровій парі DE.

За табл. 2 легко визначити 16-ричний код команди. Для цього знаходимо її в таблиці. Команда STAX D знаходиться в рядку 1 і стовпці 2. Отже, код команди STAX D – 12H.

Емулятор CPU580 дозволяє полегшити процес підбору потрібної команди з їх загального переліку. Для цього на екранній сторінці СИСТЕМА КОМАНД емулятора CPU580 потрібно в меню вибрати позицію КОМАНДЫ. В переліку типів команд, що відкрився, вибрати потрібний тип, а в переліку, що залишився, підібрати необхідну команду.

Приклад 2. Знайти команду, за допомогою якої здійснюється запис вмісту акумулятора в комірку пам'яті з заданою адресою. Оскільки за командою необхідно виконати перенесення даних, вона відноситься до команд пересилання. Оскільки в акумуляторі вміщується 1 байт інформації, то пересилання буде однобайтове. Переходимо до пошуку команди, що дозволяє виконати необхідне завдання. На головній сторінці CPU580 в меню вибираємо позицію СИСТЕМА КОМАНД; у вікні, що відкрилося, вибираємо в меню позицію КОМАНДА; в меню, що відкрилося, вибираємо ПЕРЕСЫЛОК; в підміню, що відкрилося, вибираємо ОДНОБАЙТОВЫХ. З переліку, що відкрився, відкидаємо команди MOV…, оскільки вони здійснюють міжрегістрові пересилання; MVI… – оскільки вони здійснюють запис у регістри STAX… і LDAX…, в них задіяно регістрові пари. З двох команд STA adr та LDA adr, що залишилися (дивимось у табл. 1 їх призначення), вибираємо команду **STA adr**.

2.3. Приклади складання лінійних програм

Лінійні програми характеризуються послідовним записом і виконанням команд, що входять до цих програм.

Приклад 3. Скласти програму виконання завдання: проінвертувати 16-ричні числа B4 і F6. Отримані результати скласти. Суму розмістити в комірці пам'яті з адресою 23FEH. Програму занести до ОЗП, запустити її й отримати

результат. Запуск програми виконати спочатку в автоматичному, а потім в покомандному режимі.

За табл. 1 або за допомогою емулятора (див. попередній приклад) знаходимо команду, що дозволяє виконати інвертування однобайтного числа. Це команда CMA, завдяки якій інвертується вміст акумулятора. Отже, щоб проінвертувати початкові числа, необхідно помістити спочатку одне з них до акумулятора і проінвертувати. Щоб зберегти проінвертоване число, слід перемістити його до іншого регістру, наприклад, C. Тепер до акумулятора можна помістити і проінвертувати друге число. Склавши вміст регістрів A і C, отримаємо суму проінвертованих чисел, яку можна переслати до заданої комірки пам'яті.

Підбираємо інші команди, що реалізовують виконання програми, а за табл. 2 знаходимо коди команд. Проасембльовану програму розмістимо в ОЗП, починаючи з адреси 2160H. Програма матиме вигляд, показаний в табл. 3.

Таблиця 3. Програма складання двох проінвертованих чисел і занесення суми до комірки пам'яті

Адреса	Код операції	Команда	Коментар
2160 2161	3E B3	MVI A, B3	Завантаження числа B3H до регістру A
2162	2F	CMA	Інвертування вмісту акумулятора
2163	4F	MOV C, A	Пересилання вмісту A до регістру C
2164 2165	3E F6	MVI A, F6	Завантаження числа F6H у регістр A
2166	2F	CMA	Інвертування вмісту акумулятора
2167	81	ADD C	Складання вмісту регістрів C і A. Результат – в акумуляторі
2168 2169 216A	32 FE 23	STA 23FE	Пересилка вмісту A до комірки пам'яті з адресою 23FEH
216B	76	HLT	Зупинка

Занесемо програму до ОЗП емулятора. Детально цю методику було розглянуто в лабораторній роботі МП-6, тому процес занесення програми до ОЗП покажемо без додаткових пояснень:

ОЗУ 2160 3E ВВОД B3 ВВОД 2F ВВОД... і т.д…76 ВВОД

Запуск програми в автоматичному режимі:

СБРОС РЕГ. РЕГИСТР PcH 21 ВВОД PcL 60 ВВОД ОЗУ 2160 ПОЗИЦИЯ АВТОМАТИЧЕСКИЙ РЕЖИМ ВЫПОЛНИТЬ.

В результаті виконання програми в регістрі A буде сума проінвертованих чисел (55), це ж число буде в комірці пам'яті з адресою 23FEH; у регістрі C залишиться проінвертоване перше число (4C); у лічильнику команд – адреса комірки пам'яті на одиницю більша останньої адреси програми (216CH).

Запуск програми в покомандному режимі здійснюється аналогічно, тільки при виборі режиму обирається перемикач ПОКОМАНДНЫЙ РЕЖИМ. Якщо вміст ОЗП не змінився, то етап занесення до нього програми можна опустити і відразу перейти до її запуску. Якщо змінився, необхідно занести програму знову повністю або частково, залежно від внесених змін. Покомандний режим призначений для налагодження програми і відрізняється від автоматичного тим, що тут по натисненню клавіші ВЫПОЛНИТЬ виконується лише поточна команда. Це дозволяє проглянути, як змінюється вміст регістрів в результаті її виконання. Повторне натиснення клавіші ВЫПОЛНИТЬ призводить до виконання наступної команди і так далі. В цьому режимі спрощується процес пошуку помилок в програмі й її коригування. Поетапний перегляд виконання програми краще здійснювати у вікні СТРУКТУРНАЯ СХЕМА, в якому можна простежити зміну вмісту усіх регістрів, дешифратора команд, деяких інших параметрів і сигналів в результаті виконання кожної команди. Для даного прикладу виконання програми в покомандному режимі, зміну стану регістрів і систем мікропроцесора представимо у вигляді табл. 4. Це полегшить розуміння суті виконаної окремої команди і роботи програми в цілому.

Таблиця 4. Покрокове виконання програми прикладу 3

№ кроку	Етап виконання програми	Вміст регістрів				Дешифратор команд	Буфер даних
		A	C	Регістр ознак	Лічильник команд		
0	Перед початком виконання програми	00	00	00000010	2160	–	–
1	Після виконання команди MVI A, B3 (завантаження числа B3H до регістру A)	B3	00	00000010	2162	MVI A, d8	B3
2	Після виконання команди CMA (інвертування вмісту акумулятора)	4C	00	00000010	2163	CMA	2F
3	Після виконання команди MOV C, A (пересилка вмісту A до регістру C)	4C	4C	00000010	2164	MOV C, A	4F
4	Після виконання команди MVI A, F6 (завантаження числа F 6H в регістр A)	F6	4C	00000010	2166	MVI A, d8	F6
5	Після виконання команди CMA (інвертування вмісту акумулятора)	09	4C	00000010	2167	CMA	2F
6	Після виконання команди ADD C (складання вмісту регістрів C і A. Результат – в акумуляторі)	55	4C	00010110	2168	ADD C	81
7	Після виконання команди STA 23FE (пересилка вмісту A до комірки пам'яті з адресою 23FEH)	55	4C	00010110	216B	STA adr	55
8	Після виконання команди HLT (зупинка)	55	4C	00010110	216C	HLT	76

Таблиця дозволяє відстежити результати виконання команди процесором. Так, після команди завантаження числа B3H до регістру A в акумуляторі опиниться вказане число. В наступному кроці воно буде інвертовано (команда CMA). Інверсне число (4C) залишатиметься в акумуляторі до приходу чергової команди, що змінює його вміст. У нашому прикладі це MVI A, F6 (завантаження числа F6H в регістр A - крок 4). Вміст регістра C змінюється після команди MOV C, A (пересилка вмісту A в регістр C – шаг 3) і далі залишається незмінним, оскільки команд, що впливають на регістр C, більше не зустрілося. По команді ADD C (складання вмісту регістрів C і A – крок 6) сума зберігається в акумуляторі, замість того, що було там раніше. Одночасно змінюється склад регістра ознак. До нього записуються ознаки отриманої суми: результат отримано позитивний (S = 0), не нульовий (Z = 0), містить парне число одиниць (P = 1), перенесень при складанні не було (AC = 0; C = 0). Оскільки інших команд, що впливають на регістр ознак немає, його вміст залишається незмінним до кінця програми.

У лічильнику команд завжди знаходиться початкова адреса наступної виконуваної команди. У дешифраторі команд знаходиться команда, що виконувалася і результати виконання якої записані в регістрах. У буфері даних (буферному регістрі) зберігається останній прочитаний і переданий далі байт інформації. У кроках 1…6 виконувалося читання команд з програми, розміщеної в ОЗП. Тому в цих кроках здійснювався запис у буферний регістр даних, що прийшли з ОЗП по шині даних, і потім – передача їх з буферного регістра на внутрішню шину даних. Оскільки деякі виконувані в цих кроках команди займають в ОЗП дві комірки пам'яті, тобто двобайтові, то через буфер проходить спочатку перший байт, а потім другий. Тобто після виконання команди в буфері бачимо тільки останній прочитаний і переданий далі байт. Виконувана в 7-му кроці команда (STA 23FEH) здійснює пересилання вмісту акумулятора до комірки пам'яті з адресою 23FEH. При її відпрацюванні на початку відбувається зчитування з ОЗП команди, тобто передача даних із зовнішньої шини даних через буферний регістр на внутрішню шину даних. Потім йде виконання – пересилання даних у протилежному напрямі, тобто з акумулятора до ОЗП (до комірки пам'яті з адресою 23FEH). Пересилання здійснюється ланцюжком: A → внутрішня шина даних → буферний регістр → зовнішня шина даних → ОЗП (комірка пам'яті 23FEH). Тому після виконання команди в буфері залишається останнє дане, що було передане через нього, – число 55H.

Слід зазначити, що кожна команда виконується мікропроцесором за декілька тактів і, вибравши режим ПОТАКТОВЫЙ, можна відстежити процес відпрацювання команди в кожному такті. Так, для виконання команди MVI A, B3 (крок 1) потрібно 7 тактів, команда CMA (крок 2) виконується за 5 тактів, а для виконання команди STA 23FE (крок 7) потрібно 13 тактів. Перегляд роботи в потактовому режимі виконується студентом за бажанням самостійно.

Приклад 4. Виконати завдання з п. 2.6 конспекту лекцій (табл. 2.1 і 2.2): скласти три числа, які розміщено в трьох послідовно розташованих комірках

пам'яті, та помістити результат у наступну комірку пам'яті. Доданки розміщено в пам'яті за адресами 2100H...2102H. Розглянемо обидва варіанти вирішення задачі. Оскільки початкові дані не вказані, задамося їх значеннями довільно, наприклад, 01H; 02H і 03H. Перед запуском програми занесемо ці дані до комірок пам'яті 2100H…2102H відповідно:

ОЗУ 2100 01 ВВОД 02 ВВОД 03 ВВОД

Розглянемо **1-й варіант** розв'язування задачі (табл. 2.1 з конспекту) з використанням регістрів загального призначення в якості регістрів даних.

Виконаємо асемблювання команд і розмістимо програму в ОЗП, починаючи з адреси 2180H. Оскільки в системі команд мікропроцесора КР580ВМ80 обмінюватися даними з ОЗП може тільки акумулятор, декілька змінимо програму порівняно з початковою. З урахуванням змін програма матиме вигляд, показаний в табл. 5.

Таблиця 5. Використання регістрів адреси/даних (варіант 1)

Адреса	Код операції	Команда	Коментар
2180 2181 2182	3A 02 21	LDA 2102	Помістити в A вміст комірки пам'яті з адресою 2102H (M(2102) → A)
2183	6F	MOV L, A	Переслати вміст A до регістру L (A → L)
2184 2185 2186	3A 01 21	LDA 2101	Завантажити в A вміст комірки пам'яті з адресою 2101H (M(2101) → A)
2187	67	MOV H, A	Переслати вміст A до регістру H (A → H)
2188 2189 218A	3A 00 21	LDA 2100	Завантажити в A вміст комірки пам'яті з адресою 2100H (M(2100) → A)
218B	84	ADD H	Скласти вміст регістрів A і H. Результат – в акумуляторі (A + H) → A
218C	85	ADD L	Скласти вміст регістрів A і L. Результат – в акумуляторі (A + L) → A
218D 218E 218F	32 03 21	STA 2103	Помістити вміст A до комірки пам'яті з адресою 2103H (A → M(2103))
2190	76	HLT	Зупинка

Програма зайняла в ОЗП 17 комірок пам'яті, тобто 17 байт. Підсумкова сума, отримана в результаті виконання програми, має бути рівна 06H.

Занесення програми до ОЗП:

ОЗУ 2180 3A ВВОД 02 ВВОД 21 ВВОД …і т. д.… 76 ВВОД

Запуск програми в автоматичному режимі:

СБРОС РЕГ. РЕГИСТР PcH 21 ВВОД PcL 80 ВВОД ОЗУ 2180 ПОЗИЦИЯ АВТОМАТИЧЕСКИЙ РЕЖИМ ВЫПОЛНИТЬ.

У результаті виконання програми в регістрі А буде знаходитись сума (06), те ж число буде в комірці пам'яті з адресою 2103Н; у регістрі Н – другий доданок (02); регістрі L – третій доданок (03); лічильнику команд – адреса комірки пам'яті, на одиницю більша останньої адреси програми (2191Н).

Запуск програми в покомандному режимі дозволяє проглянути, як змінюється вміст регістрів у процесі її виконання.

Розглянемо **2-й варіант** розв'язування задачі за табл. 2.2 з конспекту (з використанням регістрів загального призначення в якості регістрів адреси. Адресація здійснюється через регістрову пару HL).

Перед запуском програми занесемо початкові дані до комірок пам'яті 2100…2102:

ОЗУ 2100 01 ВВОД 02 ВВОД 03 ВВОД

Обнулимо комірку пам'яті 2103Н :

ОЗУ 2103 00 ВВОД

Виконаємо асемблювання команд і розмістимо програму в ОЗП, починаючи з адреси 21A0H. Програма набере вигляду, який показано в табл. 6.

Таблиця 6. Використання регістрів адреси/даних (варіант 2)

Адреса	Код операції	Команда	Коментар
21A0 21A1 21A2	3A 00 21	LDA 2100	Помістити в А перший операнд (вміст комірки пам'яті з адресою 2100Н) (M(2100) → A)
21A3 21A4 21A5	21 01 21	LXI 2101	Помістити в HL адресу наступного операнда (адреса комірки пам'яті 2101Н) (21 → H; 01 → L)
21A6	86	ADD M	Скласти вміст А з вмістом комірки пам'яті, адреса якої вказана в регістровій парі HL (A + M(HL) → A)
21A7	23	INX H	Інкрементувати HL (збільшити вміст регістрової пари HL на 1) (HL + 1 → HL)
21A8	86	ADD M	Скласти вміст А з вмістом комірки пам'яті, адреса якої вказана в регістровій парі HL (A + M(HL) → A)
21A9	23	INX H	Інкрементувати HL (збільшити вміст регістрової пари HL на 1) (HL + 1 → HL)
21AA	77	MOV M, A	Помістити вміст А в комірку пам'яті, вказану в регістровій парі HL (A → M(HL))
21AB	76	HLT	Зупинка

Програма зайняла в ОЗП 12 комірок пам'яті, тобто 12 байт. Підсумкова сума, отримана в результаті виконання програми, як і в попередньому варіанті, має бути рівною 06Н.

Занесення програми до ОЗП:

ОЗУ 21A0 3A ВВОД 00 ВВОД 21 ВВОД ...і т. д.... 76 ВВОД

Запуск програми в автоматичному режимі:

СБРОС РЕГ. РЕГИСТР PcH 21 ВВОД PcL A0 ВВОД ОЗУ 21A0 ПОЗИЦИЯ АВТОМАТИЧЕСКИЙ РЕЖИМ ВЫПОЛНИТЬ.

У результаті виконання програми в регістрі А буде знаходитись сума (06), те ж число буде в комірці пам'яті з адресою 2103H; у регістрі H – старший (21); регістрі L – молодший байт останньої адреси (03); лічильнику команд – адреса комірки пам'яті, на одиницю більша останньої адреси програми (21ACH).

3. Порядок виконання роботи

1. За конспектом лекцій вивчити архітектуру мікропроцесора і систему його команд.
2. Вивчити методичні вказівки до виконання цієї лабораторної роботи, розібрати наведені приклади складання програм.
3. Згідно з індивідуальним завданням скласти програму його виконання, проасемблювати її та представити у вигляді таблиці.
4. Виконати вручну розрахунки завдання у двоїчній формі. Результат перевести в 16-ричную форму. Вміст комірок пам'яті, що використовуються у програмі, приймати довільно.
5. Занести прийняті значення вмісту комірок пам'яті, що використовуються програмою в ОЗП емулятора.
6. Занести складену програму до ОЗП емулятора із заданою початковою адресою, запустити її й отримати результат. Завдання вважається виконаним, якщо результат розрахунків і результат виконання програми на емуляторі співпадають.
7. Показати результати виконання програми і виконані розрахунки викладачеві й отримати його підпис, що свідчить про виконання лабораторної роботи.

4. Зміст звіту

Звіт повинен містити:

– номер лабораторної роботи;

– найменування роботи;

– мету роботи;

– умову задачі свого варіанту;

– проасембльовану програму задачі свого варіанту, оформлену у вигляді таблиці;

– розрахунки задачі в двоїчній формі та результат у 16-ричній формі.

5. Контрольні питання

1. Скільки розрядів мають регістри загального призначення?

2 Який прийом використовується, якщо в регістрах загального призначення треба розмістити 16-розрядний операнд?

3. Скільки в мікропроцесорі серії КР580ВМ80 регістрів загального призначення?

4. Що таке стек і яке його призначення?

5. Яке призначення вказівника стека?

6. Яке призначення лічильника команд і скільки двійкових розрядів він має?

7. Зі скількох байт може складатися команда мікропроцесора?

8. Що таке мнемоніка команди і для чого вона використовується?

9. Що означає термін "операнд"? Що може бути операндом?

10. Призначення регістра ознак і які ознаки в ньому зберігаються?

11. Які функції виконують команди пересилання?

12. Які функції виконують арифметичні команди?

13. Які функції виконують логічні команди?

14. Що може бути умовою переходу до іншої частини програми?

15. Який формат має регістр ознак?

6. Рекомендована література

1. Конспект лекций по курсу „Микропроцессорная техника” для студентов направлений подготовки 6.050701 „Электротехника и электротехнологии” и 6.050702 „Электромеханика”. Части 1 и 2 / Сост. Г.М. Бажин – Днепропетровск: НГУ, каф СЭС, 2008.– 74 с.

7. Індивідуальні завдання

№ варі-анту	Почат-кова адреса	Завдання
1	2060	Скласти число 2FH з проінвертованим вмістом комірки пам'яті з адресою 2200H і від отриманої суми відняти одиницю
2	2070	Від числа FFH відняти число 05H, від результату відняти проінвертований вміст комірки пам'яті з адресою 2201H
3	2080	Від вмісту комірки пам'яті з адресою 2200H відняти вміст комірки пам'яті з адресою 2201H, результат проінвертувати
4	2090	Від вмісту комірки пам'яті з адресою 2200H відняти одиницю, результат проінвертувати і відняти від числа 18H
5	20A0	Скласти проінвертований вміст комірок пам'яті з адресами 2200H і 2201H, від результату відняти число 1CH і результат проінвертирувати

№ варі-анту	Почат-кова адреса	Завдання
6	20B0	До проінвертованого вмісту комірки пам'яті з адресою 2200H додати одиницю і від отриманого результату відняти вміст комірки пам'яті з адресою 2201H
7	20C0	Від вмісту комірки пам'яті з адресою 2200H відняти число 1EH, до результату додати одиницю й отриманий результат проінвертувати
8	20F0	До вмісту комірки пам'яті з адресою 2200H додати одиницю, проінвертувати результат і відняти від нього число 1BH
9	2100	Вміст комірки пам'яті з адресою 2200H скласти з числом 0FH і отриманий результат відняти від вмісту комірки пам'яті з адресою 2201H
10	2110	Проінвертувати вміст комірок пам'яті з адресами 2200H і 2201H, після чого скласти їх вміст і від отриманого результату відняти одиницю.
11	2120	Від суми числа OFH і проінвертованого вмісту комірки пам'яті 2201H відняти число 4EH
12	2130	Вміст комірки пам'яті з адресою 2200H проінвертувати, скласти з вмістом комірки з адресою 2201H, а потім відняти від результату число 1BH.
13	2140	Скласти два числа, розташованих у комірках пам'яті з адресами 2200H і 2201H, відняти від результату число 0AH, після чого проінвертувати отриманий результат
14	2150	Проінвертувати вміст комірки пам'яті з адресою 2200H і до результату додати одиницю, після чого від отриманого результату відняти вміст комірки пам'яті з адресою 2201H
15	2160	Проінвертований вміст комірки пам'яті з адресою 2200H скласти з різницею вмісту комірки пам'яті з адресою 2201H і числом 10H
16	2170	Скласти проінвертований вміст комірок пам'яті з адресами 2200H і 2201H, від результату відняти число 1CH і результат проінвертувати
17	2180	Скласти вміст комірки пам'яті з адресами 2200H і 2201H, від отриманого результату відняти число 1BH
18	2190	Проінвертувати вміст комірки пам'яті з адресою 2201H, додати одиницю, скласти результат з числом F0H, після чого відняти від суми число 05H.
19	21A0	Від числа FFH відняти проінвертований вміст комірки пам'яті з адресою 2200H, від результату відняти одиницю
20	21B0	Вміст комірки пам'яті з адресою 2200H скласти з різницею вмісту комірки пам'яті з адресою 2201H і числа 41H, отриманий результат проінвертувати