Міністерство освіти і науки України
Кіровоградський національний технічний університет
Кафедра автоматизації виробничих процесів

МІКРОПРОЦЕСОРНІ ЗАСОБИ ТА ЇХ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Методичні вказівки для виконання практичних робіт зі спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»

Кропивницький
2023

Міністерство освіти і науки України
Кіровоградський національний технічний університет
Кафедра автоматизації виробничих процесів

МІКРОПРОЦЕСОРНІ ЗАСОБИ ТА ЇХ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Методичні вказівки для виконання практичних робіт зі спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»

Затверджено на засіданні кафедри
Автоматизація виробничих процесів
Протокол №3 від 2022 року

Кропивницький
2023

Мікропроцесорні засоби та їх програмне забезпечення. Методичні вказівки до виконання практичних робіт зі спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» /Укл.: Трушаков Д.В., Федотова М.О. – Кропивницький: ЦНТУ, 2022 -98 с.

Укладачі: Трушаков Д.В. – канд. техн. наук, доц.
Федотова М.О. – канд. техн. наук, ас.

Рецензент: Сербул О.М. – канд. техн. наук, доц.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

На рис.1 зображено спрощену програмну модель мікропроцесора Intel I8080 в тому вигляді, в якому вона надається програмісту.

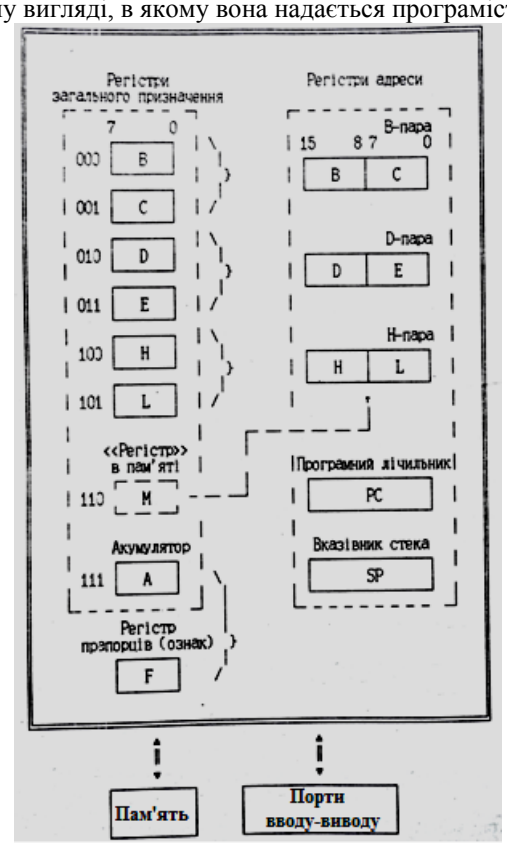


Рис. 1. Програмна модель мікропроцесора Intel I8080

Мікропроцесор Intel I8080 має:

- Акумулятор - регістр A;
- Регістри загального призначення - B, C, D, E, H, L;
- Регістр M;
- Регістрові пари - B (BC), D (DE), H (HL), PSW (AF);
- Регістр прапорців (регістр ознак) – F;
- Лічильник команд - PC;
- Регістр-вказівник стека – SP;

Регістри загального призначення - однобайтові, використовується для збереження операндів, результатів виконання операцій мікропроцесором і для організації непрямої адресації пам’яті. При виконанні деяких операцій регістри загального призначенні і регістр ознак F можуть об'єднуватися в регістрові пари: B (BC), D (DE), H (HL) і PSW (FA), як зображено на рис.2. При цьому молодшими регістрами регістрових пар є регістри C, E, L, A, а старшими - B, D, H, F.

Регістрова пара B	B	C
Регістрова пара D	D	E
Регістрова пара H	H	L
Регістрова пара PSW	A	F

Рис.2. Регістрові пари мікропроцесора Intel I8080

У регістрі ознак (прапорців) F відображаються ознаки результатів виконання операцій з даними. У цьому регістрі використовуються тільки п'ять бітів: S, Z, AC, P, C, які зображені на рис.3.

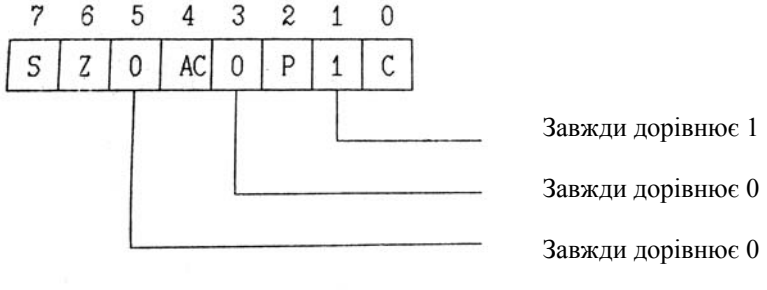


Рис.3. Регістр ознак (прапорців) мікропроцесора Intel I8080

Біти (прапорці) регістра ознак мають наступні значення:

- біт переносу (C);
- біт міжтетрадного переносу (AC);
- біт знаку (S);
- біт нуля (Z);
- біт парності (P).

Біти регістра ознак використовуються у тих командах, для яких важлива ознака результату виконання попередніх команд, наприклад, в командах передачі управління згідно умови.

Біт переносу встановлюється в "1" або "0", в результаті виконання визначених операцій. Значення цього біта можна перевірити, наприклад, за допомогою команд передачі управління за цим бітом (JC,JNC).

Якщо при виконанні операції додавання відбувається перенос з сьомого розряду регістра A (акумулятора), тоді біт переносу встановлюється в "1", якщо переносу не відбувається, то він встановлюється в "0".

Наприклад:

AE H = 1010 1110 B
+ 74 H = 0111 0100 B
122 H 1 0010 0010 B

БІТ ПЕРЕНОСУ (C) = 1

Тут символами H та B в кінці чисел помічено основу системи числення (H - шістнадцяткова, B - двійкова).

Допоміжний біт міжтетрадного переносу (AC) встановлюється в "1", якщо в результаті виконання операції відбувається перенос з третього розряду акумулятора. В іншому випадку біт міжтетрадного переносу встановлюється в "0".

Наприклад:

2E H = 0010 1110 B
+74 H = 0111 0100 B
A2 H 0 1010 0010 B

МІЖТЕТРАДНИЙ ПЕРЕНОС (AC)=1;
ПЕРЕНОС (C) = 0

Біт переносу використовується в десятковій команді корекції вмісту акумулятора (DAA).

Байт даних можна розглядати як ціле число зі знаком, який знаходиться в межах від -128 до +127. Знак числа визначає вміст старшого (сьомого) біта даних. Якщо сьомий біт дорівнює "1", то число має від'ємний знак і представлено в додатковому коді, в іншому випадку воно додатне і представлено в прямому коді. При виконанні визначених операцій біт знака встановлюється у стан, відповідний значенню сьомого розряда акумулятора, і, таким чином, може

використовуватися в командах умовного переходу для перевірки знака результату.

Біт нуля (Z) встановлюється в "1" при виконанні деяких операцій, якщо результат виконання їх дорівнює нулю. В іншому випадку цей біт встановлюється в "0".

Біт парності (P) встановлюється в "1" при виконанні ряда операцій; якщо результат виконання цих операцій містить парне число одиниць. В іншому випадку біт парності встановлюється в "0".

Наприклад:

B0 H = 1110 0000 B

+07 H = + 0000 0111B

E7 H 1110 0111 B

МІЖТЕТРАДНИЙ ПЕРЕНОС (AC) = 0;

ПЕРЕНОС (C) = 0;

БІТ ЗНАКА (S) = 0;

БІТ НУЛЯ (Z) = 0;

БІТ ПАРНОСТІ (P) =1

Лічильник команд (PC) являє собою 16-бітний регістр, доступний програмісту. Він вміщує адресу ко наш, котра виконується в даний момент часу. Після виконання команди до вмісту лічильника команд додається число, во дорівнює кількості байтів, які займає команда. Якщо виникає необхідність зміни послідовності порядку виконання команд, то в лічильник команд безпосередньо записується адреса передачі керування (вміст другого і третього байтів команди передачі керування).

Стек являє собою ділянку ОЗП. Запис в стек та читання з нього відбувається через його вершину. Адреса вершини стека визначається 16-байтовим регістром-вказником стека (SP), доступним для програміста. При записі у стек вміст регістра SP автоматично зменшується на 2, а при читанні зі стека - збільшується на 2.

Тригер переривання INTE використовується для дозволу і заборони обробки мікропроцесором запитів від зовнішіх пристроїв на переривання процесу виконання програми.

Для організаціїобміну інформацією міх мікропроцесором та зовнішніми пристроями в мікро-ЕОМ можливо використовувати до 256 портів вводу і 256 портів виводу, при цьому той самий зовнішній пристрій може мати декілька портів вводу та виводу.

Мікропроцесор серії Intel I8080 має такі зовнішні пристрої як клавіатура, принтер, адаптер для обміну даними з лінією зв'язку. Дані

передаються мікропроцесором з акумулятора в порт виводу, а приймаються з порту виводу в акумулятор.

Пам'ять являє собою масив однобайтових комірок, кожна з яких служить для зберігання команд або даних у восьмибітному двійковому коді. Порядковий номер комірки пам'яті зветься її адресою.

В мікропроцесорі серії Intel I8080 використовуються наступні типи запам'ятовуючих пристроїв:

- оперативний запам'ятовуючий пристрій (ОЗП);
- постійний запам'ятовуючий пристрій (ПЗП);
- постійний перепрограмуємий запам'ятовуючий пристрій (ППЗП).

Команди і дані можуть читатися з будь-якого типу запам'ятовуючого пристрою, а записуватися тільки в ОЗП.

У деяких командах байт пам’яті адресується подібно до регістра пам’яті мікропроцесора, тому далі такий байт буде мати назву "регістр М" адреса якого визначається вмістом регістрової пари HL.

Команди зберігаються в пам'яті мікро-ЕОМ у вигляді 8-розрядних двійкових слів і мають однобайтові, двобайтові і трибайтові формати.

Формати команд мікропроцесора серії I8080

Дані можуть зберігатися в пам'яті у вигляді окремих масивів даних або в регістрах мікропроцесора. Для звернення до команди або яо даних у пам'яті вказується адреса першого байта команди або даних.

До системи команд мікропроцесора серії Intel I8080 належать такі групи команд:

- одюбайтові пересилки типу "регістр - регістр", "пам'ять - регістр";

- двобайтові пересилки типу "регістрова пара - пам'ять", "пам'ять - регістрова пара";
- команди вводу-виводу;
- обмін двобайтовими словами;
- арифметичні і логічні операції з одним операшя;
- арифметичні і логічні операції з двома операндами;
- команди зсуву вмісту акумулятора;
- команди передачі управління;
- команди виклику і повернення з підпрограм:
- спеціальні команди.

Алгоритми виконання команд мікропроцесора подано в таблиці 1.

Таблиця 1 - Система команд мікропроцесора серії I8080

Однобайтові пересилки

```
MOV R1,R: R→R1
MVI  R8  : D8→R
STAX YZ*: A→M(YZ)
LDAX YZ*: M(YZ)→A
STA ADR : A→M(ADR)
LDA ADR : M(ADR)→A
SPHL    : HL→SP
```

Двобайтові пересилки

```
LXI YZ,D16:  D16→YZ
SHLD ADR  : HL→M(ADR)M(ADR+1)
LHLD ADR  : M(ADR)M(ADR+1)→HL
PUSH YZ** : YZ→M(SP-1)M(SP-2)
            SP-2→SP
POP YZ**  : M(SP)M(SP+1)→Y Z
(POP' PSW): SP+2→SP
```

Команди вводу-виводу

```
IN N: (N)→A
OUT N:A→(N)
```

Обмін байтами

```
XCHG :  HL↔DE
XTHL :  H↔M(SP+1), L↔M(SP)
```

Арифметичні та логічні операції з одним операндом

```
   —
CMC" : CY→CY
STC" : 1→CY
   —
CMA  : A→A
DAA' : десятк. корекція
```

```
INR''' R :   R+1 → R
DCR''' R :   R-1 → R
INX   YZ :   YZ+1 → YZ
DCX   YZ :   YZ-1 → YZ
```

Арифметичні та логічні операції з двома операндами (8-бітові операції)

```
ADD' R : A+R → A
ADC' R : A+R+CY → A
SUB' R : A-R → A
SBB' R : A-R-CY → A
ANA' R : A/\R → A
ORA' R : A\/R → A
XRA' R : A(+)R → A
```

```
ADI'D8 : A+D8 → A
ACI'D8 : A+D8+CY → A
SVI'D8 : A-D8 → A
SBI'D8 : A-D8-CY → A
ANI'D8 : A/\D8 → A
DRI'D8 : A\/D8 → A
XRI'D8 : A(+)D8 → A
```

```
CPI'D8
CMP'R
```

16-бітні операції

```
DAD"YZ : HL+YZ → HL
```

Команди зсуву змісту акумулятора

```
RLC" : зсув ліворуч-
RAL" : зсув ліворуч через біт
       ознаки  CY
RRC" : зсув праворуч
RAR" : зсув праворуч через біт
       ознаки  CY
```

Команди передачі керування

```
PCHL       : HL → PC
JMP ADR    : ADR → PC
J - CON ADR : ADR → PC
```

Спеціальні команди

```
EI  : дозволити переривання
DI  : заперечити 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

CON-(Z, NZ, C, NC, PO, PE, P, M)

Команди виклику та повернення з підпрограми

```
CALL  ADR : PC → M(SP-1)M(SP-2)
C-CON ADR : ADR → PC
RST X     : PC →  M(SP-1)M(SP-2)
          : ADR → PC, де X=0,1..7
          : ADR відповідно дорівнює
          : 0H, 8H, 10H, 18H,
          : 20H, 28H, 30H, 38H
RET       : M(SP)M(SP+1) → PC
R - CON   : SP+2 → SP
```

Умовні позначення в таблиці I

‘ - команда діє на всі ознаки: S, Z, AC, P, C; - команда діє на ознаку C;

“ - команда діє 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 регістрово! пари BC, DE, HL або регістра ЗР;

YZ* - вміст регістрової пари BC або DE;

YZ** - вміст регістрової пари BC, DE, HL або PSW;
SP - вміст покажчика стека перед виконанням команди;
D8 - 8-розрядний операнд (вміст другого байта команди);
(N) - вміст порту вводу або виводу з номером N = 0,1,...255;
D16 - 16-розрядний операнд (вміст другого та третього байтів команди);
ADR - 16-розрядна адреса у трибайтовій команді;
M(...) - вміст комірки пам'яті (адресу комірки вказано в дужках);
CON - код умови -в команді передачі керування: C0N={Z, NZ, C, NC, P0, PE, P, M};
PSW - вміст пари регістрів A і F.

Звичайно для програмування використовується дві мови - мова, якою користується програміст, або вхідна мова, і мова ЕОМ або .машинна (об'єктна) мова. Вхідна мова складається з чітко визначенного' набору символічних команд. Кожна команда має мнемонічний код, котрий нагадує програмісту операцію, яку необхідно виконати. Наприклад, символічний код операції ADD може використовуватись для позначення операції додавання.

Об'єктна мова є мовою нулів та одиниць. Вона використовується самою машиною. Для перекладу програми з мнемонічної вхідної мови на об'єктну мову (код) використовують спеціальну програму, яка має назву транслятора або асемблера.

Шістнадцяткові коди команд наведено в таблиці II.

Таблиця 2 – Шістнадцяткові коди команд мікропроцесора серії I8080

	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	0
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	1
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	2
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	3
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	4
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	5
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	6
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	7
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	8
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	9
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	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	B
C	RNZ	POP B	JNZ *	JMP *	CNZ *	PUSH B	ADI #	RST 0	RZ	RET	JZ *	–	CZ *	CALL *	ACI #	RST 1	C
D	RNC	POP D	JNC *	OUT N	CNC *	PUSH D	SUI #	RST 2	RC	–	JC *	IN N	CC *	–	SBI #	RST 3	D
E	RPO	POP H	JPO *	XTHL	CPO *	PUSH H	ANI #	RST 4	RPE	PCHL	JPE *	XCHG	CPE *	–	XRI #	RST 5	E
F	RP	POP PSW	JP *	DI	CP *	PUSH PSW	ORI #	RST 6	RM	SPHL	JM *	EI	CM *	–	CPI #	RST 7	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

N - номер порту вводу/виводу;
& - двобаштовий операнд - D16;
* - двобаштова адреса - ADR16;
# - однобайтовий операнд – D8.

Мова АСЕМБЛЕРА мікропроцесора серії І8080

Мова АСЕМБЛЕРА є машинно-орієнтованою мовою програмування мікро-ЕОМ, побудованої на основі мікропроцесора серії І8080. Він дозволяє програмісту використовувати всі функціональні можливості мікро-ЕОМ із тим самим об'ємом, як і при програмуванні на машинній мові мікропроцесора. У той же час мова АСЕМБЛЕРА звільняє програміста від трудомісткої роботи по кодуванню програми з додержанням форматів машинних команд. Крім того, мова АСЕМБЛЕРА має ряд переваг порівняно з машинною, а саме:

- символічну адресацію пам'яті;
- мнемонічне означення машинних команд;
- різноманітні способи представлення даних;
- можливість використання виразів коментування в. початковій програмі;
- умовне асемблювання, яке дозволяє змінити порядок трансляції програми;
- можливість визначення та використання в програмі макрозасобів.

Використовуючи мову АСЕМБЛЕРА можна не знижуючи ефективності програм, суттєво зменшити строки їхньої розробки та налагодження порівняно з випадком застосування машинної мови. У той же час мова АСЕМБЛЕРА поступається мовам більм високого рівня (БЕЙСІК, ФОРТРАН. ПАСКАЛЬ) за трудомісткістю побудови та налагодження програм.

Програма, яка написана мовою АСЕМБЛЕРА, будується з послідовного запису речень на мові АСЕМБЛЕРА. Всі речення на мові АСЕМБЛЕР можна поділити на три типи:

- пусте речення;
- речення макровизначення;
- оператор мови АСЕМБЛЕРА.

Якщо програма на машинній мові записана на будь-якому носії інформації, то її можна завантажити в пам'ять. Така програма зветься ***об'єктною***. Приклад трансляції початкової програми зображено на рис.4.

Рис.4. Трансляція початкової програми до об'єктної

Речення початкової програми записується за допомогою символів, які являють собою абетку мови АСЕМБЛЕР:

- літери латинської, російської, те української абетки;
- цифри 0, 1, 2, 3. 4, 5, 6. 7. 8. 9.

Символічні імена

Символічне ім'я - це послідовність літер та цифр, які починаються з літери. Довжина послідовності обмежується, але різними вважаються символічні імена, які відрізняються першими п'ятьма символами.

В мові АСЕМБЛЕРА використовують два типи символічних імен:

- постійні;
- імена, які визначаються користувачем.

Постійні символічні імена не потребують визначення до використання їх у початковій програмі, оскільки вони, зберігаються у спеціальній таблиці АСЕМБЛЕРА, яка має назву таблиці постійних символічних імен. До цієї таблиці записуються:

- мнемокоди машинних команд;
- мнемокоди псевдокоманд ORG, IF, ENDIF, END, DB, DW, DS, EQU, SET, MACRO, END;
- імена регістрів та регістрових пар A, B, C, D, E, H, L, PSW, PC, SP, M.

Символічні імена користувача повинні обов'язково бути визначені в початковій програмі. Під час трансляції вони виділяються з початкової програми і записуються до таблиці символічних імен, визначених користувачем. Символічні імена, визначені користувачем, неповинні співпадати з постійними. В іншому випадку буде подано діагностичну інформацію про помилку.

Система команд мікропроцесора серії I8080

Сукупність команд, які виконуються мікропроцесором, створює систему команд. За характером операцій їх прийнято розділяти на групи команд:

- передача даних між пристроями системи;
- арифметичні і логічні операції;
- команди передачі управління і виклику підпрограм (команди переходу);
- спеціальні команди.

При вивченні системи команд мікропроцесора слід звернути увагу їхнє кодування, зміст, спосіб адресації, довжину, час виконання.

Далі розглянуто команди, які входять до системи команд мікропроцесора серії І8080. Слід враховувати, що при програмуванні на машинній мові команди, як і дані, подаються двійковим кодом. Однак в такому вигляді програма погано сприймається людиною, тому краще складати програму, використовуючи символічне позначення кодів операцій, регістрів, адрес комірок пам'яті. При вводі в мікропроцесор символічні позначення необхідно замінити кодами. При кодуванні команд, які утримують в полі операнда адресу комірки пам'яті, слід вказувати молодший байт адреси в другому байті команди, а старший - в третьому.

ПРАКТИЧНА РОБОТА 1

Мета робота: 1. Оволодіти методикою переведення чисел з однієї системи числення в іншу.
2. Виконати індивідуальне завдання.

1. КОРОТКІ ВІДОМОСТІ

1.1. Форма зображення чисел

ЕОМ містить велику кількість комірок пам'яті 1 регістрів (латинню registrum - внесене, записане) для зберігання двійкової Інформації. Більшість цих комірок має однакову довжину і, тобто вони використовуються для зберігання і біт двійкової інформації. Інформація, що знаходиться в такій комірці, *зветься* словом. Слово 16-бітової ЕОМ, зображено на рис. 1.1.

Зручна для використання в ЕОМ двійкова система числення зовсім не зручна для запису та читання чисел людиною. Дійсно, замість, наприклад, чотиризначного десяткового числа 8769 доводиться працювати з його 14-розрядним еквівалентом - двійковим числом 1000100100001. Але, до цього приходять лише при спілкуванні з ЕОМ на рівні її машинної мови. Однак у теперішній час така мова часто використовується при роботі з мікропроцесорами та мікро-ЕОМ.

Для зменшення трудомісткості ручної обробки кодів чисел та команд використовують вісімкову і шістнадцяткову системи числення. У вісімковій системі числення використовується 8 цифр (0,1,2,3,4,5,6,7), в шістнадцятковій - 10 цифр і 6 великих латинських букв (0,1,2,3.4,5,6,7,8,9, A,B,C,D,E,F).

1.2. Системи числення та їхні перетворення

Одне й те саме число можна записати по-різному у різних системах числення. У цифровій техніці широко використовують такі системи числення: двійкову, вісімкову, десяткову, шістнадцяткову. Будь-яка система числення має певний набір цифр і правил їхнього записування. Загальну кількість усіх цифр системи називають II основою. Поряд написані цифри утворюють число. У конкретному числі кожна цифра займає певну позицію. Якщо та сама цифра має різне значення залежно від позиції, то систему числення називають позиційною. Розглянемо, наприклад, число 666, записане в десятковій системі числення. Цифра 6 повторюється в числі три рази, однак щоразу вона має інше значення. У крайній справа Позиції ця цифра означає шість одиниць, у наступній - шість десятків, в останній - шість сотень. Отже, десяткова система числення - позиційна. Двійкова, вісімкова, шістнадцяткова системи числення також є позиційними.

Крім позиційних, є непозиційні системи числення, наприклад римська. У числі XXX цифра X записана в різних позиціях, але кожного разу означає одну й ту саму величину - десять одиниць.

Узагальнена формула, за якою записують числа в будь-якій позиційній системі числення, має вигляд:

$$K = \sum_{i=-m}^{i=n-1} a \cdot r^i$$

де К – число;

a - цифри, які використано в певній системі числення;

r - основа системы числення;

i - біжучий індекс, який змінюється від -m до n-1, де n - кількість розрядів числа, що стоять зліва від коми;

m - кількість розрядів числа справа від коми.

З використанням наведеної вище формули запишемо такі числа:

$125.3_{(10)} = 1 * 10^2 + 2 * 10^1 + 5 * 10^0 + 3 * 10^{-1}$,

$212.2_{(3)} = 2 * 3^2 + 1 * 3^1 + 2 * 3^0 + 2 * 3^{-1}$,

$237.5_{(8)} = 2 * 8^2 + 3 * 8^1 + 7 * 8^0 + 5 * 8^{-1}$,

$964_{(16)} = 9 * 16^2 + 6 * 16^1 + 4 * 16^0$,

Треба пам'ятати, що цифри, використані в системі числення, залежать від основи цієї системи, причому молодшою цифрою завжди є 0, старшою - цифра, на одиницю менша від основи системи числення. Наприклад, цифри, які використовують у десятковій системі числення: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; у двійковій: 0, 1; у трійковій: 0, 1, 2; у вісімковій: 0, 1, 2, 3, 4, 5, 6, 7; у шістнадцятковій: 0, 1, 2..., 14, 15.

Якщо число записують у шістнадцятковій системі, яку широко використовують, то двоцифрові числа 10, 11,..., 15 позначають у горі спеціальними знаками, наприклад:

$$\overset{\Lambda}{1}\ \overset{\Lambda}{10}\ 12\ 5_{(16)} = 1 * 16^3 + 10 * 16^2 + 12 * 16^1 + 5 * 16^0 \ ,$$

У цифровій техніці для спрощення записів шістнадцяткових чисел, двоцифровим числам 10, 11, 12, 13, 14, 15 присвоюють такі буквені позначення: 10 - A, 11 – B, 12 - C, 13 - D, 14 - E, 15 - F.

Тоді шістнадцяткове число $\overset{\Lambda}{1}\ \overset{\Lambda}{10}\ 12\ 5_{(16)}$

можна записати скорочено: $1AC5_{(16)}$.

Існує декілька методів перетворення чисел з однієї системи числення в іншу.

Часто основи двійкової і шістнадцяткової систем числення позначають великими латинськими буквами: двійкову - B (binary), шістнадцяткову - H (hexadecimal).

Метод додавання заміщених величин

Для переведення числа з системи з основою k в систему з основою r необхідно зобразити число в розгорнутому вигляді в системі з основою к, всі цифри в основі к замінити на їхні зображення в системі з основою r, виконати всі операції множення і додавання в системі з основою r.

Приклад 1.1

Переведення числа з двійкової системи числення $10111010_{(2)}$ в десяткову систему числення.

$$10111010_{(2)} = 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 186_{(10)}$$

Приклад 1.2

Переведення числа з шістнадцяткової системи числення $A28{,}8_{(16)}$ в десяткову систему числення.

$$A28{,}8_{(16)} = A \cdot 16^2 + 2 \cdot 16^1 + 8 \cdot 16^0 + 8 \cdot 16^{-1}$$

$$= 10 \cdot 256 + 2 \cdot 16 + 8 \cdot 1 + 8/16 = 2600{,}5$$

Метод ділення цілих чисел на основу

У практичній роботі часто буває потрібно перетворювати числа із однієї системи числення в числа будь-якої іншої системи числення. Для цього користуються правилом ділення десяткового числа на основу тієї системи числення, в яку треба перетворити дане число.

Для цього ціле число однієї системи поділимо на основу бажаної іншої системи числення. Значення остачі присвоюємо молодшому розряду шуканого числа бажаної іншої системи. Результат першого ділення знову ділимо на основу бажаної іншої системи числення. Остачі присвоюємо наступний розряд числа бажаної нової системи числення. Подібну процедуру повторюють доти, поки результат ділення дорівнюватиме 0. Остача від останньої дії ділення є значенням старшого розряду числа бажаної нової системи числення.

Приклад 1.3

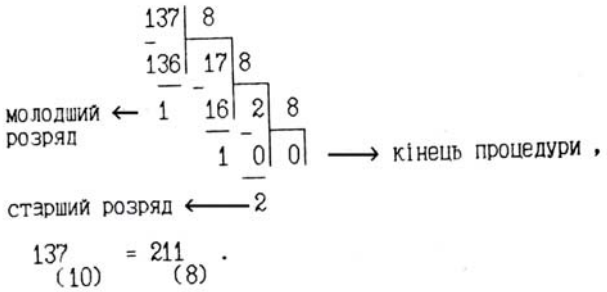
Переведення числа з десяткової системи числення в двійкову систему числення.

Для цього ціле десяткове число поділимо на основу двійкової системи числення, тобто на 2. Остача від ділення на 2 може дорівнювати 0 або 1. Значення остачі присвоюємо молодшому розряду шуканого двійкового числа. Результат першого ділення знову ділимо на 2. Остачі (знову 0 або 1) присвоюємо наступний розряд двійкового числа. Подібну процедуру повторюють доти, поки результат ділення дорівнюватиме 0. Остача від останньої дії ділення є значенням старшого розряду двійкового числа.

Отже, результатом перетворення десяткового числа 24 у систему числення є число 11000.

Приклад 1.4

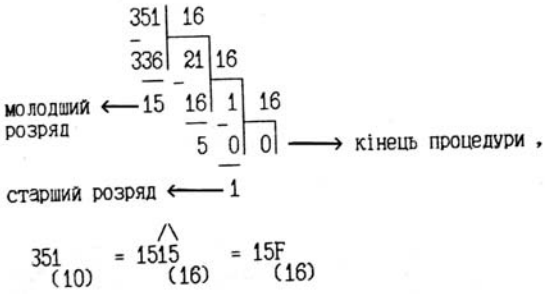
Записати десяткове число 137 у вісімковій системі числення.
Процедура перетворення матиме такий вигляд:



$137_{(10)} = 211_{(8)}$.

Приклад 1.5

Записати десяткове число 351 у шістнадцятковій системі числення. Процедура перетворення матиме такий вигляд:



$351_{(10)} = 1515_{(16)} = 15F_{(16)}$

Метод множення дробу на основу

Щоб перетворити десятковий дріб у двійкову систему числення, треба скористатися іншим правилом. У цьому разі дробову частину десяткового числа множать на основу двійкової системи числення, тобто на 2. Якщо результат буде менший від 1, то старшому розряду шуканої дробової частини двійкового числа присвоюють значення 0, а якщо більший від 1, то 1. Результат попередньої операції знову множать на 2, причому для множення беруть його дробову частину. Аналогічно роблять з новим результатом. Описану процедуру

повторюють доти, поки результат наступного множення точно не дорівнюватиме 1 або поки не буде досягнуто потрібної точності.

Приклад 1.6

Перетворення десяткового числа 0,625 в двійкову систему числення.

$0.625 \cdot 2 = 1.25 \longrightarrow$ 1 - старший розряд дробової частини двійкового числа,

$0.25 \cdot 2 = 0.5 \longrightarrow$ 0 - наступний розряд дробової частини двійкового числа ,

$0.5 \cdot 2 = 1 \longrightarrow$ 1 - молодший розряд дробової частини двійкового числа ,

$0.625_{(10)} = 0.101_{(2)}$.

Приклад 1.7

Переведення числа з десяткової системи числення в двійкову систему числення.

$0.6875_{(10)}$

*	2
1	3750
*	2
0	7500
*	2
1	5000
*	2
1	0000

$0.6875_{(10)} = 0.1011_{(2)}$

Приклад 1.8

Переведення числа з десяткової системи числення в шістнадцяткову систему числення.

$0.6875_{(10)}$

*	*16*
4	*1250*
+	
6	*875*
11	*0000*

$0.6875_{(10)} = 0.B_{(16)}$

Отже, якщо десяткове число з дробовою частиною треба перетворити в двійкову систему числення, то цілу його частину перетворюють за першим правилом, а дробову - за другим 1 потім записують загальний результат.

При переведенні невірного дробу цілу частину переводять методом ділення, а дробову - множення, результати об'єднують.

Переведення двійкових чисел у шістнадцяткову систему числення та у зворотному напрямку

Основа шістнадцяткової системи числення (16) являє собою основу двійкової системи числення (2), взяту в четвертому степені.

Тобто $16 = 2^4$.

Виходячи з цього, одну цифру шістнадцяткової системи числення можна виразити в чотирьох розрядах двійкової системи числення.

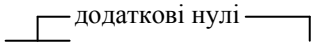
Для переведення чисел з двійкової системи числення в шістнадцяткову систему числення достатньо цифрові частини двійкового числа справа і зліва від коми розбити на тетради (по чотири двійкових розряди), та додати до старшої 1 молодшої тетради недістаючі нулі (до чотирьох розрядів), кожній двійковій тетраді поставити у відповідність шістнадцяткову цифру з таблиці 1.

Приклад 1.9

Перевести число К з двійкової системи числення в шістнадцяткову систему числення.

$$К = 110001100,1110001_{(2)} .$$

Розбиваємо число К, починаючи від коми справа, на тетради, додаючи до старшої і молодшої тетради недістаючі нулі.



$$К = \underline{0001}\ \underline{1000}\ \underline{1100}, \underline{1110}\ \underline{0010}\ .{}_{(2)}$$

Замінюємо кожне значення двійкової тетради шістнадцятковим еквівалентом і отримуємо число К в шістнадцятковій системі числення.

$К = 18C,E2_{(16)}$.

При переведенні чисел з шістнадцяткової системи числення в двійкову систему числення, кожну цифру числа слід замінити відповідним їй двійковим еквівалентом, який складається з чотирьох розрядів.

Приклад 1.10

Переведення числа К із шістнадцяткової системи числення в двійкову систему числення.

$$К = CF,2B_{(16)} = 11001111,00101011_{(2)} .$$

Переведення двійкових чисел у вісімкову систему числення та у зворотному напрямку

Основа вісі косової системи числення (8) являє собою основу двійкової системи числення (2), взяту в третьому степені.

Тобто $8 = 2^3$.

Виходячи з цього, одну цифру вісімкової системи числення можна виразити в трьох розрядах двійкової системи числення.

Для переведення чисел з двійкової системи в вісімкову систему числення достатньо цифрові частини двійкового числа справа і зліва від коми розбити на триади (по три двійкових розряди), та додати до старшої і молодшої триади недостаючі нулі (до трьох розрядів), кожній двійковій триаді поставити у відповідність вісімкову цифру з таблиці 1.

Приклад 1.11

Перевести число К з двійкової системи числення в вісімкову систему числення.

$$К = 11000110,1110001_{(2)} .$$

Розбиваємо число К, починаючи від коми праворуч на триади, додаючи до старшої і молодшої триади недостаючі нулі,

додаткові нулі

$$К = \underline{110}\ \underline{001}\ \underline{110},\ \underline{111}\ \underline{000}\ \underline{100}$$

Замінюємо кожне значення двійкової триади вісімковим еквівалентом і записуємо число К у вісімковій системі числення.

$$К=110001100,1110001_{(2)} = 614,704_{(8)} .$$

При переведенні чисел із вісімкової системи числення у двійкову систему числення, кожну цифру числа слід замінити відповідним їй двійковим еквівалентом, який складається з трьох розрядів.

Приклад 1.12

Перевести число К із вісімкової системи числення у двійкову систему числення.

$$К = 710,614_{(8)} = 111001000,110001100_{(2)} .$$

Таблиця 1 - Порівняння десяткових, двійкових, вісімкових і шістнадцяткових кодів чисел

Десяткові числа	Двійкові числа	Вісімкові числа	Шістнадцяткові числа
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13

20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F
32	100000	40	20

ДВІЙКОВА АРИФМЕТИКА

Додавання, віднімання, множення і ділення двійкових чисел

Над двійковими числами, як і над числами, записаними у будь-якій іншій системі числення, можна виконувати арифметичні операції: додавання, віднімання, множення 1 ділення.

Додавання і віднімання двійкових чисел подібні до таких самих дій над десятковими числами.

Дію додавання розпочинають з додавання цифр молодших розрядів доданків. Якщо результат додавання більший від 1, то до наступного розряду переносять одиницю, а в молодшому розряді пишуть 0. Потім додають цифри наступних розрядів з урахуванням одиниць, перенесених з попереднього розряду до одержання шукано! суми.

Таблицця додавання двікових чисел має такий вигляд:

0+0=0 ,
0+1=1 ,
1+0=1 ,
1+1=10 .

Як видно з таблиці, додавання двох одиниць дає 0 у наймолодшому розряді, а одиниця переноситься в наступний старший розряд.

Приклад 1.13

Додати два двійкових числа: 10001 і 10111.

1 0 1 1 1 - перенесення
1 0 0 0 1 - перший доданок
+
1 0 1 1 1 - другий доданок
1 0 1 0 0 0 - сума

Дію віднімання розпочинають із наймолодших розрядів. Якщо який-небудь із розрядів двійкового числа зменшуваного дорівнює 0, а однойменний розряд від'ємника дорівнює 1, то позичають 1 з сусіднього старшого розряду зменшуваного. Якщо позичено одиницю старшого розряду, то в сусідньому молодшому розряді матимемо дві одиниці.

Таблиця віднімання двійкових чисел має такий вигляд:

0 - 0 = 0,
1 - 0 = 1 ,
1 - 1 = 0,
10-1 = 1 .

Приклад 1.14

Відняти від двійкового числа 1010 двійкове число 101.

-1010
101
0101

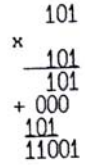
Множення двійкових чисел виконують за правилами, аналогічними для десяткових чисел, тобто визначають проміжні добутки, а потім їх додають.

Таблиця множення двійкових чисел має такий вигляд:

1 x 1 = 1 ,
0 x 1 = 0 ,
1 x 0 = 0 ,
0 x 0 = 0 .

Приклад 1.15

Помножити двійкове число 101 на двійкове число

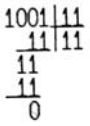


Отже, операція множення двійкових чисел зводиться до операцій зміщення і додавання. Тому в цифрових пристроях для множення використано схеми додавання і зміщення.

Ділять двійкові числа, як і десяткові, у відповідності з правилами множення і віднімання двійкових чисел.

Приклад 1.16

Поділити двійкове число 1001 на двійкове число 11.



ПОНЯТТЯ ПРО КОДУВАННЯ

Прямий, зворотний і додатковий двійковий коди. Додатні і від'ємні десяткові числа у двійковому коді. Двійково-десяткові коди

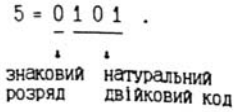
Позначення різної інформації відповідними - цифрами називають цифровим кодуванням, а послідовність цифр для певної інформації - її кодом. Описану вище систему запису довільного десяткового числа за допомогою двох цифр 0 та 1 називають натуральним двійковим кодом.

У цифровій електроніці для спрощення деяких математичних операцій часто використовують певні двійкові коди, які дістають від перетворення натурального двійкового коду. Це прямий, зворотний і додатковий коди.

Прямі коди використовують для записування додатних і. від'ємних чисел. Лоб знайти прямий код, треба до натурального двійкового коду дописати зліва від десяткового числа ще один розряд, який називають знаковим. Якщо число буде додатним, то знаковим розрядом буде 0, якщо від'ємним - 1. Під час записування прямого коду знаковий розряд окреслюють інтервалом.

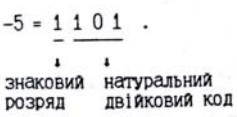
Приклад 1.17

Записати число 5 прямим кодом.



Приклад 1.18

Записати число -5 прямим кодом.



Від'ємні числа часто записують у зворотному і додатковому кодах, що спрощує побудову деяких цифрових пристроїв.

Щоб знайти зворотний код від'ємного десяткового числа, треба спочатку записати його прямим кодом, а потім одиниці змінити на нулі і нулі на одиниці в усіх розрядах, крім знакового.

Приклад 1.19

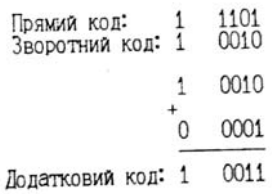
Знайти зворотний код числа -28.

Прямий код: 1 11100
Зворотний код: 1 00011

Щоб знайти додатковий код від'ємного додаткового числа, треба до його зворотного коду додати 1.

Приклад 1.20

Знайти додатковий код числа -13.



Використавши додаткові коди, можна замінити, наприклад, операцію віднімання на операцію додавання прямого коду зменшуваного з додатковим кодом від’ємника. Така заміна дає

можливість використати один електронний пристрій, який називають суматором, як для додавання, так і для віднімання двійкових чисел.

Приклад 1.21

Відняти від числа 7 число 5.
Прямий код зменшуваного (7): 0 111
Прямий код від'ємника (-5): 1 101
Зворотний код від'ємника (-5): 1 010

$$\begin{array}{r} + \; 1\,010 \\ 0\,001 \\ \hline 1\,011 \end{array}$$

Додатковий код:

Додамо прямий код зменшуваного до додаткового коду від'ємника. Знакові розряди беруть участь у додаванні як старші розряди. Якщо переносять одиницю із знакових розрядів, то її відкидають:

$$\begin{array}{r} + \; 0\,111 \\ 1\,011 \\ \hline 10\,010 \end{array}$$

відкидається

Остаточний результат 0 010 = $1 \times 2^1 = 2$

знаковий розряд

Крім описаних вище кодів, у цифрових пристроях широко використовують двійково-десяткові коди, їх також можна використовувати для записування десяткових чисел за допомогою 0 та 1.

Кожна окремо взята десяткова цифра в такому коді записується своїм двійковим, чотирьохрозрядним еквівалентом. Якщо, наприклад, треба у двійково-десятковому коді записати число 75, то кожну цифру в цьому числі треба окремо перевести у двійковий код і потім записати загальний результат:

$7 = 0110_{(2)}$;
$5 = 0101_{(2)}$;
$75 = 0110\ 0101_{(2\text{-}10)}$.

Слід зазначити, що, коли, наприклад, десяткову цифру перетворюють у натуральний двійковий код, то дістають трьохрозрядне число $110_{(2)}$, проте за правилами утворення двійково-десяткових кодів треба знайти чотирьохрозрядне двійкове число, а

тому зліва дописують 0; на значення числа така операція не впливає. Те саме стосується й інших десяткових цифр крім 8 та 9, які при переведенні в натуральний двійковий код одразу дають чотирьохрозрядний результат.

Не можна плутати правила утворення двійково-десяткового коду і натурального двійкового коду. Щоб знайти натуральний двійковий код числа 75, все це число переводять згідно з розглянутими вище правилами в двійкову систему і дістають такий результат:

$75 = 1001011_{(2)}$

2. ЗАВДАННЯ НА САМОСТІЙНУ РОБОТУ

1. З числами, які задає викладач, виконати наступні операції.
   - 1.1 За методом додавання заміщених величин перевести числа з двійкової, вісімкової, і шістнадцятих) во І систем числення у десяткову систему числення.
   - 1.2 За методом ділення цілих чисел на основу перевести числа з десяткової системи числення у двійкову, вісімкову, і шістнадцяткову системи числення.
   - 1.3 За методом множення дробу на основу перетворити десятковий дріб у двійкову і шістнадцяткову системи числення.
   - 1.4 Перевести число з двійкової системи числення у шістнадцяткову систему числення та навпаки.
   - 1.5 Перевести число з двійкової системи числення у вісімкову систему числення та навпаки.
2. Виконати операції двійкової арифметики з числами, які задає викладач.
   - 2.1 Скласти два двійкових числа.
   - 2.2 Відняти двійкові числа.
   - 2.3 Помножити двійкові числа.
   - 2.4 Поділити двійкові числа.
3. Виконати наступні операції кодування з десятковими числами, які задає викладач.
   - 3.1 Записати ПРЯМИМ, зворотним і додатковим кодами.
   - 3.2 Записати двійково-десятковим кодом.
   - 3.3 Скориставшись дією додавання прямого коду зменшуваного з додатковим кодом від'ємника, виконати операцію віднімання.

ПРАКТИЧНА РОБОТА 2

КОНТРОЛЕР ПРОГРАМОВАНИЙ УНІВЕРСАЛЬНИЙ (КПУ)
"ЕЛЕКТРОНІКА МС 2702" ПРИЗНАЧЕННЯ І СКЛАДОВІ ЧАСТИНИ
ОСНОВНІ ФУНКЦІЇ МОНІТОРА

Мета роботи : 1. Вивчення складу та основних функцій КПУ
2. Вивчення команд МОНІТОРА КПУ

1. КОРОТКІ ВІДОМОСТІ

1.1. КПУ "ЕЛЕКТРОНІКА МС 2702". Призначення і складові частини

КПУ відноситься до мікропроцесорних засобів обчислювальної техніки загального призначення і може бути використаним у складі систем керування технологічним обладнанням, у складі випробувального, контрольновимірювального та іншого обладнання.

КПУ виконаний на основі великих інтегральних схем (ВІС) серії І8080 (КР580), К589, К565, КР556, К568 та інтегральних схем (ІС) серії К155. Базовим комплектом є ВІС серії КР580.

З елементів переліченої серії в КПУ використано такі ВІС:

а) І8080 (КР580ИК80А) – однокристальний восьмирозрядний паралельний мікропроцесор;

б) КР580ИК51 - універсальний синхронно-асинхронний програмований прийомо-передавач;

в) КР580ИК55 - програмований пристрій вводу-виводу паралельної інформації;

г) КР580ВН59 – програмований контролер переривань;

д) КР580ВИ53 - програмований інтервальний таймер;

е) К568РЕ1 - масочно-програмований ПЗП (постійний запам'ятовуючий пристрій) ємністю 2 Кбайти;

ж) К573РФ1 - ППЗП (перепрограмований постійний запам'ятовуючий пристрій) ємністю 512 байтів;

з) К573РФ1 - ППЗП з електричним програмуванням і ультрафіолетовим стиранням ємністю 1 Кбайт;

і) К573РФ2 - ППЗП з електричним програмуванням і ультрафіолетовим стиранням ємністю 2 Кбайт;

к) К589АП26 - шинний формувач інвертуючий;

л) К589АП16 - шинний формувач;

м) К589ИР12 - багаторежимний буферний регістр;

н) КР580ГФ24 – генератор тактових імпульсів.

ПГВП ППЗП з електричним програмуванням і стиранням ємністю 2 Кбайти; шинний Формувач інвертуючий; шинний Формувач;

багаторежимний буферний регістр; генератор тактових імпульсів.

Пристрій КПУ

Функціональні частини КПУ - контролер і пульт керування (ПК), виконані у вигляді самостійних конструктивів.

Клавіатуру ПК набрано на основі клавіш ВМ-27-1-1.

ВІС серії І8080 (КР580), ВІС ЗП, встановлюються на спеціальних колодках. Це забезпечує зручність при виконанні технологічних операцій контролю і при ремонті контролера. На контактних колодках встановлюється також ППЗП, оскільки користувач повинен програмувати ЗП самостійно. Крім того, користувачу надано право вибору або зміни типу ЗП, і це право забезпечується в наслідку використання конструктивних елементів (штире-контактних) для перекомутації зв'язку.

Розподіл адрес поля пам'яті і адрес поля пристроїв вводу/виводу подано в таблиці і. Пам'ять контролера складається з пристроїв пам'яті і буферів даних для обміну з магістраллю даних.

Пристрій пам'яті містить 1 КБАЙТ ОЗП (оперативний запам'ятовуючий пристрій), 4 КБАЙТ ППЗП, 4 КБАЙТ ПЗП,а також ВІС ЗП для зберігання програм користувача (встановлюється за індивідуальним замовленням).

Позиційне позначення елемента ЗП	Поле адрес	Примітка
D16	0000-07FF РФ2	Ініціалізація, тести, бібліотека 4К
D17	0800-0FFF РФ2	Бібліотека 4К
	1000-17FF РЕ1	
	1800-1FFF РЕ1	
D4 - D11	2000-23FF РУ2	ОЗП Адреси 2000-2060 зайняті оперативними даними МОНІТОРА, адреси 2061-20FF відведені під стек, адреси 2100-23FF - для ОЗП користувача 1К
	2800-2FFF	
D6 пульт	3000-37FF РФ2	МОНІТОР 4К
	3800-3FFF	
	4000-47FF	
	4800-4FFF	
D25	5000-57FF РФ2	ППЗП за індивідуальним замовленням
D24	5800-5FFF РФ2	ППЗ за індивідуальним замовленням
D23	6000-67FF РФ2	ППЗП за індивідуальним замовленням
D22	6800-6FFF РФ2	ППЗП за індивідуальним замовленням
D21	7000-77FF РФ2	ППЗП за індивідуальним замовленням
D20	7800-7FFF РФ2	ППЗП за індивідуальним замовленням

1.2. Технічні дані

КПУ має наступні технічні характеристики:

- тип використовуваного мікропроцесора - паралельний;
- розрядність паралельно оброблюваної Інформації - вісім двійкових розрядів;
- форма зображення чисел - додатковий код Із фіксованою комою; є можливість використання арифметичних операцій, тригонометричних, показникових і логарифмічних функцій в діапазоні

зміни чисел від $0,5867.10^{-38}$ до $0,1704.10^{39}$ за допомогою бібліотеки підпрограм з плаваючою комою;

- пряма адресація до пам'яті ємністю до 64 Кбайт;
- види адресації - неявна, пряма, непряма, безпосередня (адресованою одиницею є байт);
- максимальне число адресованих зовнішніх пристроїв: 256 - вводу і 256 - виводу:
- кількість команд - 78, включаючи команди арифметичних 1 логічних операцій, операцій десяткової корекції, стекові операції, складання слів двійкової довжини, операції вводу/виводу, операції керування;
- формат команд - однобайтна, двобайтна, трибайтна;
- час виконання команд типу регістр-регістр - не більше 2 мкс;
- система переривання пріоритетна, з програмним встановленням пріоритету, режимів обслуговування 1 масок, з шістьма запитами на переривання при роботі ПК 1 вісьма запитами на переривання при роботі без ПК, з можливістю збільшення кількості запитів переривання до 24;
- ємність ОЗП - 1 Кбайт;
- ємність ПЗП - змінний параметр;
- ємність ППЗП - змінний параметр;
- загальне число програмованих паралельних каналів вводу/виводу Інформації - 48, в тому числі тих, що мають гальванічну розв'язку, -8 (4- для вводу інформації, 4 - для виводу інформації);
- кількість програмованих послідовних каналів вводу інформації - 1, виводу інформації - 1;
- кількість 18-розрядних програмованих інтервальних таймерів, які використовують в якості багаторежимного задаючого елемента: 2 - при роботі з ПК і 3 - при роботі без ПК;
- діалог з контролером відбувається за допомогою програмованого ПК і керуючої програми МОНІТОР;
- обмін інформацією між ПК і контролером - програмно-апаратний в режимі реального часу;
- налагодження програм користувача можливе за допомогою керуючої програми МОНІТОР:
- контроль працездатності основних вузлів КПУ виконується за допомогою тест-програм;
- введення (ручний набір) інформації виконується за допомогою 16 клавіш ("0"- "F")ПК;

- введення (ручний набір) операцій керування - трьома клавішами ПК ("[_]", "CR", "#");
- число позицій індикованих цифр (знаків) та букв латинського алфавіту - 9;
- організація одночасного (візуального) індикування необхідних сегментів цифр (знаків), букв виконується програмними засобами;
- готовність КПУ до роботи після подачі напруги живлення або сигналу "СКИД" практично миттєва;
- режим експлуатації КПУ безперервний, або періодичний з багаторазовим вмиканням/вимиканням напруги живлення протягом доби;
- можливе підключення до контролера допоміжних ЗП (запам'ятовуючих пристроїв) із прямою адресацією до 64 Кбайт і пристроїв вводу/виводу до 256;
- можливий перерозподіл зон пам'яті контролера і допоміжних ЗП, які підключає користувач;
- режим роботи КПУ - автоматичний, при налагодженні програм є можливість виконання програм у крокovому режимі;
- ємність ППЗП в ПК, який застосовується для зберігання програм МОНІТОРА, - 2 Кбайти:
- встановлення КПУ і системи користувача в початковий стан виконується вручну за допомогою клавіші "СКИД" на ПК і автоматично після подачі напруги живлення.

Габаритні розміри корпусу контролера не більші від 358*244*57 мм, ПК - 237*100*40 мм.

Маса контролера не більша від 1.5 кг, пульта керування - не більша від 0,8 кг.

Споживаюча потужність КПУ не більша 35 Вт. Струм, який споживається КПУ по напругах +5В, -5В, +12В, не більше 5А, 0.1А, 0.4А відповідно.

1.3. МОНІТОР КПУ "Електроніка МС 2702"

Функції

МОНІТОР КПУ "Електроніка МС 2702" призначено для керування виконанням програм КПУ і налагодження їх.

МОНІТОР у режимі реального часу виконує наступні функції:

а) керування виконанням програм:

б) керування обміном інформації з зовнішніми пристроями:

в) відліку програм.

Розподіл пам'яті

Програма МОНІТОР КПУ займає об'єм 4 Кбайти і записана в ППЗП пульта за адресами 300H-37FFH і 000H-07FFH.

МОНІТОР КПУ використовує в якості робочих комірок ОЗП за адресами 2000H-2060H.

Для забезпечення виконання програми МОНІТОР КПУ необхідно ПК підключити до контролера.

Характеристики програми

Звертання до програми МОНІТОР КПУ передбачає її початковий запуск. Для запуску програми необхідно:

а) ввімкнути живлення;

б) натиснути клавішу “R” ;

в) натиснути клавішу ”#”. У крайній лівій позиції індикатора (в позиції 1) символ “–“ показує готовність системи до роботи.

Принцип організації

МОНІТОР КПУ являє собою модуль діалогової взаємодії користувача з КПУ. Діалог "КПУ - користувач" виконується з пульта КПУ. При завданні з пульта КПУ наступної команди МОНІТОРА виконується переривання працюючої програми і перехід на обробку відповідної команди.

Користувач звертається до МОНІТОРА за допомогою команд МОНІТОРА.

Дія МОНІТОРА зводиться до обробки його команд. Коли закінчиться виконання команди МОНІТОРА, буде виконуватись передача керування на виконання перерваної програми.

Індикацію організовано в динамічному режимі з видачею вмісту визначеної області пам'яті з інтервалом 2 мс за сигналами переривання, які надходять з таймера.

Процес вводу команди, а також результат її виконання відображається на дев'яти цифрових індикаторах, які знаходяться на передній панелі пульта КПУ.

Перед початком вводу кожної команди на пульті КПУ натискають клавішу "#". Якщо натиснути цю клавішу, відбувається установка в нуль відповідних комірок ОЗП, буфера, тобто виконуються підготовчі операції для виконання будь-якої команди. При цьому у крайній лівій позиції індикатора з'являється символ "-", потім вводиться ім'я команди.

При цьому виконуються підготовчі операції - готуються визначенні константи, які необхідні для виконання даної команди.

При вводі імені команди символ "-" у крайній лівій позиції індикатора замінюється ім'ям команди.

Якщо команда вимагає вводу параметрів, то у крайній правій позиції індикатора з'явиться цифра "1", яка засвідчує те, що повинен вводитися перший параметр. Потім, в залежності від типу команди, вводяться параметри команди: однобайтні або двобайтні через розділювачі їв якості розділювача КПУ призначено клавішу "пробіл"), або передається керування на виконання програми, коли натиснути клавішу “CR”, команди (E,S) починають виконуватись після натиснення клавіші “пробіл”.

При вводі параметрів у крайній правій позиції індикатора індикується номер параметра, який вводиться.

При вводі цифр параметра кожна нова цифра вводиться в молодший розряд поля, відведеного під індикацію параметра, а попередня інформація зсувається на одну позицію ліворуч. Цифри, які виходять зліва за формат параметра, губляться.

Тому, якщо в процесі вводу параметра зроблено помилку, можна не скидати команду (за допомогою клавіші "#"), а продовжувати ввод до того часу, поки на індикаторі не з'явиться потрібний параметр.

Параметри, які вводяться з пульта, надходять у пам'ять після натискання клавіші "пробіл" (крім кінцевого, який надходить після натискання клавіші "CR").

При невиконанні послідовності вводу команди у всіх позиціях індикатора індикується символ "r", який засвідчує те, що команду набрано з пульта невірно.

В залежності від форми виконання команди можливі три режими роботи програми МОНІТОР:

1. Робота на фоні програм користувача

В цьому режимі після обробки кожного символа, який вводиться з пульта, керування передається на програму користувача. Ознакою переходу на програму користувача є символ "-" в позиції 9 дисплея при закінченні виконання команди. При переході на програму користувача за командою G без точок розриву відключається динамічна індикація.

2. Автономний режим

Режим встановлюється при виконанні команди G з точками розриву. В цьому режимі працює тільки МОНІТОР. Після обробки наступного символу виконується зупинка. Режим відміняється за

командою б без точок розриву. Ознакою роботи в автономному режимі є символ "=" в позиції 9 дисплея по закінченні виконання команди.

3. Режим трасування (кроковий режим)

Режим, аналогічний автономному. При виконанні програми за командою J здійснюється зупинка після кожної команди програми, що трасується (дивись команду J). Цей режим встановлюється командою J і відміняється командою відміни трасування A. Ознакою роботи в режимі трасування є символ "три горизонтальних риски" в позиції 9 індикатора після закінчення виконання команди.

Команди МОНІТОРА

Систему команд МОНІТОРА складають 12 команд: C, D, E, F, G, S, I, O, J, A, P, B. Команди за типом виконуваної підрозділяються на три групи:

а) команда керування виконанням програми (G):
б) команди забезпечення відладки (E, S, J, A):
в) сервісні команди (F, C, D, I, O, P, B).

Команда C

За командою C (мнемоніка COMPARE - порівняти) виконується порівняння області пам'яті, обмеженої першими двома параметрами, з областю пам'яті, починаючи з третього параметра команди.

При непорівнянні двох областей пам'яті на індикатор виводиться адреса комірки із першої області. Порівняння продовжується після натискання клавіші "пробіл" (продовження).

Приклад 1

Треба порівняти вміст області пам'яті , обмежений адресами 2100H-2110H, зі вмістом області пам'яті, починаючи з адреси 2120H.

Нижче наведено послідовність набору команди

C 2100 [_] 2110 [_] 2120 CR

у вигляді послідовності натиснення клавіш та видачі повідомлень на індикаторі, як подано в таблиці 2.

При цьому попарно порівнюється вміст комірок 2100 і 2120, 2101 і 2121, і т.д.

У випадку, якщо знайдено невідповідність (непорівняння), на індикатор видається адреса відповідної комірки з першої області і процес порівняння зупиняється. Для продовження необхідно натиснути "пробіл".

Таблиця 2

Клавіша	Індикація									Примітки
	1	2	3	4	5	6	7	8	9	
#	-									Встановлення пульта в початковий стан
C	C								1	Ввід команди
2	C					2			1	Ввід першого параметра
1	C				2	1			1	
0	C			2	1	0			1	
0	C		2	1	0	0			1	
[_]	C								2	
2	C					2			2	Ввід другого параметра
1	C				2	1			2	
1	C			2	1	1			2	
0	C		2	1	1	0			2	
[_]	C								3	
2	C					2			3	Ввід третього параметра
1	C				2	1			3	
2	C			2	1	2			3	
0	C		2	1	2	0				
CR	C									Передача керування на виконання
	C		2	1	0	0				Непорівняння вмісту за адресою 2100H
	C									Команду виконано

Команда D

За командою D (мнемоніка DISPLAY – індикувати) виконується індикація вмісту комірки пам'яті з адреси, вказаної в команді.

За цією командою можна візуально спостерігати зміни вмісту вказаної комірки пам'яті при виконанні програми.

Приклад 2

Треба індикувати вміст комірки пам' яті з адресою 2276H.

Нижче наведено послідовність набору команди # D 2276 CR у вигляді послідовності натиснення клавіш та видачі індикаторі, як подано в таблиці 3.

В процесі виконання команди в позиціях 7, висвітлюється (в шістнадцятковому коді) вміст комірки Якщо процесор зайнятий в цей час виконанням робочої цієї комірки може мінятися в момент його перезапису.

Таблиця 3

Клавіша	Індикація									Примітки
	1	2	3	4	5	6	7	8	9	
#										Встановлення пульта в початковий стан
D	D									Ввід команди
2	D				2					Ввід параметра
2	D			2	2					
7	D		2	2	7					
6	D	2	2	7	6					
CR	D	2	2	7	6			0	1	
	D	2	2	7	6			0	1	
	D	2	2	7	6			0	2	
	D	2	2	7	6			0	1	
	D	2	2	7	6			0	1	
	D	2	2	7	6			0	3	

Команда E

За допомогою команди E (мнемоніка EMBARK - завантажувати) виконується дослідження і модифікація вмісту регістрів програми користувача:

A - акумулятор;

B, C, D,E,H,L - регістри загального призначення мікропроцесора;

F - регістр ознак мікропроцесора;

P - організований в оперативній пам'яті 16-бітовий регістр адреси передачі керування користувача;

S - показник стека.

Виконання програми виконується в наступному режимі:

а) вводиться символ Е, а потім розділювач "пробіл" (ім'я регістра виводиться в позиції 3 індикатора);

б) в позиціях 5-6 (або в позиціях 5-8) виводиться вміст досліджуваного регістру;

в) при вводі розділювача "пробіл" на індикатор виводиться ім'я наступного регістра та його вміст;

г) модифікація поточного регістра виконується шляхом вводу нового вмісту (у шістнадцятковому вигляді), а після нього - розділювач "пробіл";

д) при вводі символу "#" дослідження вмісту регістрів припиняється,' виконується перехід на продовження перерваної команди.

Приклад 3

Треба індикувати вміст регістрів.

Нижче наведено послідовність набору команди

E [_] [_] [_] [_] [_] [_] [_] [_] [_] [_]

і індикація вмісту регістрів A, B, C, D, E, H, I, L, F, P, S у вигляді послідовності натиснення клавіш та видачі повідомлень, на індикаторі, як подано в таблиці 4.

Таблиця 4

Клавіша	Індикація									Примітки
	1	2	3	4	5	6	7	8	9	
#	-									Встановлення пульта в початковий стан
E	E									Ввід команди
[_]	E		A		7	8				Індикація вмісту регістра A
[_]	E		B		A	F				Індикація вмісту регістра B
[_]	E		C		D	3				Індикація вмісту регістра C
[_]	E		D		F	F				Індикація вмісту регістра D
[_]	E		E		D	D				Індикація вмісту регістра E
[_]	E		F		0	S				Індикація вмісту регістра F
[_]	E		H		A	1				Індикація вмісту регістра H
[_]	E		L		7	0				Індикація вмісту регістра L
[_]	E		P		0	2	3	4		Індикація вмісту регістра P
[_]	E		S		5	0	3	4		Індикація вмісту регістра S

[_]	E		A		7	8				Індикація вмісту регістра A
[_]	E		B		A	F				Індикація вмісту регістра B
#	-									Встановлення пульта в початковий стан

Приклад 4

Треба замінити вміст регістрів A і D.
Нижче наведено послідовність набору команди
E [_] 24 [_] [_] [_] 12 [_]
у вигляді послідовності натиснення клавіш та видачі повідомлень на індикаторі, як подано в таблиці 5.
Таблиця 5

Клавіша	Індикація									Примітки
	1	2	3	4	5	6	7	8	9	
#	-									Встановлення пульта в початковий стан
E	E									Ввід команди
[_]	E		A		7	8				Індикація вмісту регістра F
2	E		A		8	2				Вміст регістра D=FFH замінюється на D=12H
4	E		A		2	4				
[_]	E		B		A	F				Індикація вмісту регістра P
[_]	E		C		D	3				Індикація вмісту регістра S
[_]	E		D		F	F				Індикація вмісту регістра A
1	--		D		F	1				Вміст регістра D=FFH замінюється на D=12H
2	E		D		1	2				
[_]	E		E		0	0				Індикація вмісту регістра E
[_]	E		F		0	5				Індикація вмісту регістра F
[_]	E		H		A	1				Індикація вмісту регістра H
[_]	E		L		7	0				Індикація вмісту регістра L
[_]	E		P		0	2	3	4		Індикація вмісту регістра P
[_]	E		S		5	0	3	4		Індикація вмісту регістра S
[_]	E		A		2	4				Індикація вмісту регістра A
[_]	E		B		A	F				Індикація вмісту регістра B
#	--									Встановлення пульта в початковий стан

При цьому вміст регістра A буде замінено на A=24H, а регістра D – на D=12H.

Команда F

За командою F (мнемоніка FILL - заповнити) виконується заповненя константи області оперативної пам'яті. Перший і другий параметри команди являють собою відповідно початкову і кінцеву адреси заповнюваної області пам'яті. Третій параметр є заповнюючою константою.

Приклад 5

Треба заповнити область пам'яті з адресами 2140H-2354H константою, що дорівнює ABH.

Нижче наведено послідовність набору команди

F 2140 [_] 2354 [_] AB CR

у вигляді послідовності натиснення клавіш індикаторі, як подано в таблиці 6.

Таблиця 6

Клавіша	Індикація									Примітки
	1	2	3	4	5	6	7	8	9	
#	-									Встановлення пульта в початковий стан
F	F								1	Ввід команди
2	F					2			1	Ввід першого параметра
1	F				2	1				
4	F			2	1	4			1	
0	F		2	1	4	0			1	
[_]	F								2	
2	F					2			2	Ввід другого параметра
3	F				2	3			2	
5	F			2	3	5			2	
4	F		2	3	5	4			2	
[_]	F								3	
A	F					A			3	Ввід третього параметра
B	F				A	B			3	
CR	F									Передача керування на виконання

Виконання можна перевірити за допомогою команди D.

ПРАКТИЧНА РОБОТА 3

КОНТРОЛЕР ПРОГРАМОВАНИЙ УНІВЕРСАЛЬНИЙ (КПУ)
"ЕЛЕКТРОНІКА МС 2721"
ПРИЗНАЧЕННЯ І СКЛАДОВІ ЧАСТИНИ
ОСНОВНІ ФУНКЦІЇ МОНІТОРА

Мета роботи : 1. Вивчення складу та основних функцій КПУ
2. Вивчення команд МОНІТОРА КПУ

1. КОРОТКІ ВІДОМОСТІ

КПУ "ЕЛЕКТРОНІКА МС 2721". Призначення і складові частини

КПУ відноситься до мікропроцесорних засобів обчислювальної техніки загального призначення і може бути використаним у складі систем керування технологічним обладнанням, у складі випробувального, контрольновимірювального та іншого обладнання.

КПУ виконаний на основі великих інтегральних схем (ВІС) серії КР580, К589, К573, К555, К531, К170, К293, К537 та інтегральних схем (ІС) серії К155. Базовим комплектом є ВІС серії КР580.

З елементів переліченої серії в КПУ використані такі ВІС:

а) І8080 (КР580ВМ80А) - однокристальний восьмирозрядний паралельний мікропроцесор;

б) КР580ВВ51А - універсальний синхронно-асинхронний програмований прийомо-передавач;

в) КР580ВВ55А - програмований пристрій вводу-виводу паралельної інформації;

г) КР580ВВ79 - програмований контролер клавіатури дисплея;

д) КР1810ВН59А - програмований контролер переривань;

е) КР580ГФ24 - генератор тактових імпульсів;

ж) КР580ВИ53 - програмований універсальний таймер;

з) К580ВА86 - восьмирозрядний шинний формувач;

и) К580ВА87 - восьмирозрядний шинний формувач інвертуючий;

к) КР537РУ8А - ОЗП (оперативний запам'ятовуючий пристрій) ємністю 16 Кбіт (2Кх8);

л) К573РФ5 - РПЗП (репрограмований постійний запам'ятовуючий пристрій) з ультрафіолетовим стиранням ємністю 16 Кбіт (2Кх8);

м) К573РФ6А - РПЗП (репрограмований постійний запам'ятовуючий пристрій) з ультрафіолетовим стиранням ємністю 64 Кбіт (8Кх8), який використовується в модифікаціях КПУ;

н) К170АП2 - 2 формувача сигналів для лінії зв'язку апаратури передачі даних;

о) К170УП2 - 4 підсилювача сигналів для лінії зв’язку апаратури передачі даних.

Пристрій КПУ

Функціональні частини КПУ - контролер і пульт керування (ПК) - виконані у вигляді самостійних конструктивів.

Клавіатуру ПК набрано на основі клавіш ВМ-27-1-1.

ВІС серії 580, ВІС ЗП встановлюються на спеціальних колодках. Це забезпечує зручність при виконанні технологічних операцій контролю і при ремонті контролера. На контактних колодках встановлюється також РПЗП, оскільки користувач повинен програмувати ЗП самостійно. Крім того користувачу надано право вибору або зміни типу ЗП, і це право забезпечується в наслідку використання конструктивних елементів (штире-контактних) для перекомутації зв'язку.

Пам'ять контролера складається з пристроїв пам'яті і буферів данних для обміну з магістраллю даних.

Пристрій пам'яті містить 2 Кбайти ОЗП (оперативний запам'ятовуючий пристрій), 4 Кбайти РПЗП, 4 Кбайти ПЗП, а також ВІС ЗП для зберігання програм користувача (встановлюється за індивідуальним замовленням).

2. ХАРАКТЕРИСТИКИ ПРОГРАМИ

2.1. Розподіл пам'яті

Програма МОНІТОР КПУ має об'єм F2EH байтів і записана в РПЗП на платі пристрою центрального процесора за адресами 0000H – 0F2EH.

Оперативний запам'ятовуючий пристрій (ОЗП) має адреси: 1800H - FFFH (2 Кб), адреси користувача: 18A0 – 1FFFH.

2.2. Принцип організації

Програма МОНІТОР КПУ є модулем діалогової взаємодії користувача з КПУ. Діалог "КПУ - користувач" відбувається за допомогою пульта КПУ. При надходженні з пульта КПУ чергової команди програми МОНІТОР відбувається переривання працюючої програми користувача і здійснюється перехід до обробки заданої команди. Користувач звертається до програми МОНІТОР за допомогою відповідних команд.

Дія програми МОНІТОР зводиться до обробки її команд. Після закінчення виконання команди програми МОНІТОР керування передається перерваній програмі.

В залежності від форми виконання команди можливі три режими роботи програми МОНІТОР:

1) робота на фоні програми користувача. В цьому режимі після обробки кожного символа, який вводиться з пульта, і після закінчення виконання програми МОНІТОР (крім команд G і ST) керування передається програмі користувача.

2) автономний режим. Режим встановлюється при досягненні однієї з точок розриву, встановленої командою G з точками розриву. В цьому режимі виконується тільки програма МОНІТОР. Програму користувача зупинена на точці розриву. Режим відміняється за командою G без точок розриву, таким чином передається керування програмі користувача:

3) режим трасування. Цей режим встановлюється командою ST. Після кожної виконаної команди відбувається зупинка програми і керування передається програмі МОНІТОР, таким чином можливий ввід будь-якої команди, крім G та ST. Режим продовжується командою ST і відміняється командою G без точок розриву.

Індикацію здійснює мікросхема КР580ВВ79.

Процес вводу команди, а також результат її виконання . відображаються на восьми цифрових індикаторах, розташованих на передній панелі пульта КПУ. Символ в крайній лівій позиції індикатора свідчить про готовність програми МОНІТОР до прийняття команди.

При вводі імені команди символ « – » в позиції 1 індикатора заміняється символами « , »в позиціях 1-4 або 2-3 в залежності від формату введеної команди. Символи, які висвітлюються у вигляді « , », говорять про необхідність вводу параметрів команди, якщо це необхідно користувачеві. Параметри команд вводяться через відокремлював «[_]».

Команда починає виконуватись при натисканні клавіші «CR». Деякі команди, наприклад X, S, починають виконуватися при натисканні клавіші «[_]» і закінчуються при натисканні клавіші «CR». При вводі недостатнього або надлишкового числа параметрів на індикаторі висвітлюється "Errз". Список можливих помилок подано у розділі 6. При вводі цифр параметра кожна нова цифра вводиться в молодший розряд поля, відведеного під індикацію параметра, а попередня інформація зсувається на одну позицію ліворуч. Цифри, які виходять уліво за формат параметра, втрачаються.

Тому, якщо в процесі вводу параметра припущено помилку, можна не скидати команду (за допомогою клавіші "#"), а продовжувати ввід, доки на індикаторі не з'явиться потрібний параметр. Параметри, що вводяться з

пульта, надходять у пам’ять після натиснення клавіші «[_]», а останній з них вводиться натисканням клавіші «CR».

3. ЗВЕРТАННЯ ДО ПРОГРАМИ

3.1. Звертання до програми МОНІТОР КПУ передбачає її початковий запуск. Для запуску програми необхідно:

1) пульт керування підключити до пристрою центрального процесора;

2) ввімкнути живлення;

3) натиснути клавішу « R ». При цьому на індикаторі відображується -80 1.0 ;

4) натиснути клавішу «#» При натисненні цієї клавіші відбувається початкова установка відповідник комірок ОЗП, таким чином здійснюються підготовчі операції для виконання команд програми МОНІТОР. При цьому в крайній лівій позиції індикатора висвітлюється символ " - ", що показує готовність системи до робота. Якщо в пам'яті є програма, то їй автоматично передається керування.

4. ВХІДНІ ТА ВИХІДНІ ДАНІ

4.1. Команди монітора

Систему команд монітора складають 8 команд:

S, X, G, ST, I, O, M, C.

Команди за типом виконуваної ними функції поділяються на групи:

27 команда керування запуском програми (G);

28 команди забезпечення налагодження (S, X, ST);

29 сервісні команди (C, I, 0, M);

30 команди запуску тестів (TI, T2, T3).

33 Команда C

За командою C (мнемоніка COMPARE - порівняти) виробляється порівняння вмісту області пам’яті, відокремленої першими двома параметрами, зі вмістом області пам'яті, яка починається з третього параметра команди.

При непорівнянні вмісту двом областей пам'яті на індикатор видається адреса комірки пам’яті першої з них. Порівняння продовжується після натиснення клавіші "CR”.

Приклад 1

Вимагається порівняти вміст області пам’яті, відокремленої адресами 0100H - 0102H, зі вмістом області пам’яті, починаючи з адреси 0104H.

Нижче подано послідовність набору команди

C 0100 [_] 0102 [_] 0104 CR CR CR

у вигляді послідовності натиснення клавіш та видачі повідомлень на індикаторі, як показано в таблиці

Таблиця 1

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
#									Встановлення пульта в початковий стан
C		У	У	У			1		Ввід команди C
0	У	У	У	0			1		Ввід першого параметра
1	У	У	0	1			1		
0	У	0	1	0			1		
0	0	1	0	0			1		
[_]	У	У	У	У			2		
0	У	У	У	0			2		Ввід другого параметра
1	У	У	0	1			2		
0	У	0	1	0			2		
2	0	1	0	2			2		
[_]	У	У	У	У			3		
0	У	У	У	0			3		Ввід третього параметра
1	У	У	0	1			3		
0	У	0	1	0			3		
4	0	1	0	4			3		
CR	0	1	0	1					Непорівняння вмісту за адресою 0101H
CR	0	1	0	2					Непорівняння вмісту за адресою 0102H
CR									Команду виконано

4.1.2. Команда X

За допомогою команди X (EXAMINE - дослідити) виконується дослідження і модифікація вмісту регістрів програми користувача:

A - акумулятор;

B,C, D,E,H,L - регістри загального призначення мікропроцесора;

F - регістр ознак мікропроцесора;

P - іб-бітовий регістр адреси передачі керування програмі користувача. При зупинках в точках розриву регістр P має значення регістра адреси переривання програми користувача;

S- вказівник стеку.

Виконання команди виконується у такій послідовності:

1) вводиться символ X;

2) в позиції 2 індикатора виводиться ім'я регістра A, в позиціях 5-6 - вміст регістра, що досліджується;

3) при вводі дільника « [_] » на індикатор виводиться ім'я наступного регістра та його вміст (в позиціях 5 - 6 або 4-7);

4) модифікація вмісту даного регістра виконується шляхом вводу з пульта нового вмісту (в шістнадцятковому коді), а після нього – дільника« [_] » ;

5) при натисканні клавіши «CR»дослідження вмісту регістрів завершується.

Приклад 2

Необхідно індикувати вміст регістрів і замінити вміст регістра B на число 12H.

Нижче подано послідовність набору команди

X [_] [_] [_] [_] [_] [_] [_] [_] [_] [_] [_] 12 CR

у вигляді послідовності натиснення клавіш та видачі повідомлень на індикаторі, як показано в таблиці 2.

Таблиця 2

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
#									Встановлення пульта в початковий стан
X		A			X	X			Ввід команди X
[_]		B			X	X			
[_]		C			X	X			
[_]		D			X	X			

[_]		E			X	X			
[_]		H			X	X			
[_]		L			X	X			
[_]		F			X	X			
[_]		P		X	X	X	X		
[_]		S		X	X	X	X		
[_]		A			X	X			
[_]		B			X	X			Індикація вмісту регістра B і заміна його вмісту на число 12H
1					X	1			
2					1	2			
CR	-								Команду виконано

Примітка : X означає, що вміст регістрів не визначено однозначно.

4.1.3. Команда S

За командою S (мнемоніка SUBSTITUTE - замінити) виконується дослідження і модифікація вмісту комірок пам'яті.

Виконання команди здійснюється у діалоговому режимі. Послідовність виконання команди наступна:

1) вводиться символ S ішістнадцяткова адреса першої досліджуваної комірки пам’яті, а потім роздільник « [_] ». При цьому адреса комірки виводиться в позиціях 1-4, вміст досліджуваної комірки пам'яті в позиціях 6 - 7 індикатора;

2) виконується модифікація поточної комірки пам'яті шляхом вводу з пульта нового змісту (в шістнадцятковому коді), а потім вводиться РОЗДІЛЬНИК « [_] ».

3) при натисненні клавіші "CR" дослідження вмісту комірок пам'яті припиняється.

Приклад 3

Проконтролювати вміст комірок пам'яті 1A0 0H - 1A0 2 H і модифікувати вміст комірки 1A 0 1H.

Нижче подано послідовність набору команди

S 1A0 0 [_] [_] 1A [_] C R

у вигляді послідовності натиснення клавіш та видачі повідомлень на індикаторі, як показано в таблиці 3.

Таблиця 3

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
#	-								Встановлення пульта в початковий стан
S	У	У	У	У					Ввід команди S
1	У	У	У	1					Ввід адреси комірки пам’яті
A	У	У	1	A					
0	У	1	A	0					
0	1	A	0	0					
[_]	1	A	0	0		X	X		Індикація вмісту комірок пам’яті за адресами 1 A0 0H і 1A0
[_]	1	A	0	1		X	X		
1	1	A	0	1		X	1		Модифікація вмісту комірки пам’яті за адресою 1A01H
A	1	A	0	1		1	A		
[_]	1	A	0 .	2		X	X		
C R	-								Команду виконано

Примітка: X означає, що вміст комірок пам'яті не визначено однозначно.

4.1.4. Команда M

Команда M переміщує вміст однієї області пам’яті в іншу. Дані початкової області залишаються без змін. Виконання команди відбувається у наступній послідовності:

1) вводиться символ M і шістнадцяткова адреса початку вихідної області пам’яті, тобто адреса першого байта, що треба перемістити. При цьому адреса комірки виводиться в позиціях 1-4 індикатора;

2)вводиться роздільник « [_] »а потім шістнадцяткова адреса останнього байта, що треба перемістити, тобто кінця початкової області пам'яті;

3)вводиться роздільник « [_] »,а потім шістнадцяткова адреса початку області призначення;

4)при натисненні клавіші "CR" відбувається виконання команди, а по її закінченню індикується символ "_" в першій позиції індикатора.

Приклад 4

Перемістити дані з області пам'яті з адресами 1A00H-1A02H в область з початковою адресою 1A05H.

Нижче подано послідовність набору команди
M 1A00 [_] 1A02 [_] 1A05 CR
у вигляді послідовності натиснення клавіш та видачі повідомлень на індикаторі, як показано в таблиці 4

Таблиця 4

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
#	-								Встановлення пульта в початковий стан
M		9	9	9			1		Ввід команди М
1	9	9	9	1			1		Ввід початкової адреси початкової області пам' яті
A	*	9	1	A			1		
0		1	A	0			1		
0	1	A	0	0			1		
[_]	9	9	9	9			2		
1	9	9	9	1			2		Ввід кінцевої адреси початкової області пам’яті
A	9	9	1	A			2		
0	9	1	A	0			2		
2		A	0	2			2		
[_]	9	9	. 9	9			3		
1	9	9	9	1			3		Ввід адреси пам'яті області призначення
A	9	9	1	A			3		
0	9	1	A	0			3		
5	1	A	0	5			3		
CR	-								Команду виконано

4.1.5 Команда I

За командою I (мнемоніка INPUT - ввід) виконується індикація вмісту регістра зовнішнього пристрою за адресою, що вводиться з пульта.

Приклад 5
Вимагається проконтролювати вміст регістра зовнішнього пристрою за адресою F4H.
Нижче подано послідовність набору команди

I F4 [_] CR

у вигляді послідовності натиснення клавіш та видачі повідомлень на індикаторі, як показано в таблиці 5.

Таблиця 5

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
#									Встановлення пульта в початковий стан
I		*9* .	*9*						Ввід команди I
F			F						Ввід адреси регістра зовнішнього пристрою
4			4						
[_]		F	4			X	X		Індикування вмісту регістра зовнішнього пристрою
CR	-								Команду виконано

Примітка: X означає, що вміст комірок пам'яті не визначено однозначно.

4.1.6 Команда 0

4.1.7

За командою 0 (мнемоніка OUTPUT - вивід) виконується вивід байта даних на регістр зовнішнього пристрою за адресою, що вводиться з пульта.

Приклад 6

Вимагається вивести на регістр зовнішнього пристрою за адресою F4H число 55H.

Нижче подано послідовність набору команди

O P4 [_] 55H CR

у вигляді послідовності натиснення клавіш та видачі повідомлень на індикаторі, як показано в таблиці 6.

Таблиця 6

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
#	-								Встановлення пульта в початковий стан
0		*У*	*У*						Ввід команди 0
P		*у*	F						Ввід адреси регістра зовнішнього пристрою
4		F	4						

[_]		F	4			У	У		
5		F	4			У	5		Ввід числа 55Н
5		F	4			5	5		
CR	-								Команду виконано

4.1.7. Команда G

Команда G (мнемоніка GO - йти) є командою запуску програми. Запуск програми полягає у здійсненні необхідних передач керування від програми МОНІТОР до виконуваної програми і навпаки. Команда G має декілька режимів.

Якщо заданий тільки перший параметр, він інтерпретується як точка входу в програму користувача і відбувається передача керування за цією адресою.

Якщо задані додаткові параметри (один чи два), вони розглядаються як точки розриву, тобто адреси програм, при досягненні якої керування передається програмі МОНІТОР. При цьому стан перерваної програми запам'ятовується і на індикатор виводиться адреса точки розриву. При натисненні клавіші « # » індикується символ « _ » що свідчить про готовність програми МОНІТОР до сприймання команд.

Якщо перший параметр не заданий (або замість нього введений роздільник ”[_]"), збережене значення лічильника команд програми користувача застосовується як точка входу в цю програму. _ Отже, відсутність першого параметра означає, що програма, яка виконувалась останньою, запускається з адреси попередньої точки розриву. При налагодженні програм в реальному масштабі часу необхідно враховувати слідуючу обставину. Оскільки реалізація команди *G з* точками розриву залежить від стану лічильника таймера, після кожної виконаної команди з подальшою її обробкою відбувається суттєве уповільнення виконання заданої частини програми (від точки входу до точки розриву).

Якщо команда G без параметрів або введений тільки перший з них (точки розриву не вказані), програма виконується в реальному масштабі часу.

Приклад 2.1

Введемо у пам’ять наступну послідовність команд

MVI A,04H

MVI B, 07H

DCR B

JMP 1C00H

за адресою 1A00H у шістнадцятковому коді за допомогою команди "S"
S 1A00 [_] 3E [_] 04 [_1 06 [_1 07 [_] 05 [_] C3 [_] 00 [_] 1C CR
Ця послідовність команд у пам'яті у шістнадцятковому коді буде мати вигляд, як показано в таблиці 7.1.
Таблиця 7.1

Адреса	Вміст
1A00	3E
1A01	04
1A02	06
1A03	07
1A04	05
1A05	C3
1A06	00
1A07	1C

Нижче подано послідовність набору команди у вигляді послідовності натиснення клавіш та видачі повідомлень на індикаторі, як показано в
Таблиця 7.2

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
#	-								Встановлення пульта в початковий стан
S	У	*	У	У					Ввід команди S
1	у	У	У	1					Ввід адреси комірки пам’яті
A	у	У	1	A					
0	У	1	A	0					
0	1	A	0	0					
[_]	1	A	0	0		X	X		Індикація вмісту комірки пам’яті
3	1	A	0	0		X	3		Модифікація вмісту комірки пам’яті за адресою 1A00H
E	1	A	0	0		3	E		

[_]	1	A	0	1		X	X		
0	1	A	0	1		X	0		Модифікація вмісту комірки пам’яті за адресою 1A01H
4	1	A	0	1		0	4		
[_]	1	A	0	2		X	X		
0	1	A	0	2		X	0		Модифікація вмісту комірки пам'яті за адресою 1A02H
6	1	A	0	2		0	6		
[_]	1	A	0	3		X	X		
0	1	A	0	3		X	0		Модифікація вмісту комірки пам'яті за адресою 1A03H
7		A	0	3		0	7		
[_]	1	A	0	4		X	X		
0	1	A	0	4		X	0		Модифікація вмісту комірки пам'яті за адресою 1A04H
5	1	A	0	4		0	5		
[_]	1	A	0	5		X	X		
C	i	A	0	5		X	C		Модифікація вмісту комірки пам’яті за адресою 1A05H
3	i	A	0	5		C	3		
[_]	1	A	0	6		X	X		
0		A	0	6		X	0		Модифікація вмісту комірки пам'яті за адресою 1A06H
0	1	A	0	6		0	0		
[_]	1	A	0	7		X	X		
1	1	A	0	7		X	1		Модифікація вмісту комірки пам˙яті за адресою 1A07H
C	1	A	0	7		1	C		
[_]	1	A	0	8		X	X		
CR	-								Команду виконано

ПРИМІТКА: X означає, що вміст комірок пам'яті не визначено однозначно.

Передамо керування за адресою 1A00H і задамо точки розриву за адресами 1A02H, 1A04H. Нижче подано послідовність набору команди

G 1A00 [_] 1A02 [_] 1A04 CR

у вигляді послідовності натиснення клавіш та видачі повідомлень на індикаторі, як показано в таблиці 7.3.

Таблиця 7.3

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
#	-								Встановлення пульта в початковий стан
G	9	9	9	9			1		Ввід команди G
1	9	9	9	1			1		Ввід першого параметра
A	9	9	1	A			1		
0	9	1	A	0			1		
0	1	A	0	0			1		
[_]	9	9	9	9			2		
1	9	9	9	1			2		Ввід другого параметра
A	9	9	1	A			2		
0	9	1	A	0			2		
2		A.	0	2			2		
[_]	9	9	9	9			3		
1	9	9	9	1			3		Ввід третього параметра
A	9	9	1	A			3		
0	9	1	A	0			3		
4	1	A	0	5			3		
CR	-								Відбулася зупинка за адресою 1A02H. Зміст пам'яті 06H

Приклад 7.2

Необхідно виконати програму» починаючи з попередньої точки розриву, тобто з адреси 1A02H. Послідовність натиснення клавіш і видачі повідомлень на індикаторі показано в таблиці 7.4.

Таблиця 7.4

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
#	-								Встановлення пульта в початковий стан
G	9	9	9	9			1		Ввід команди G
[_]	9	9	9	9			2		
CR	-								Команду виконано

4.1.8. Команда ST

При вводі команди БТ (мнемоніка STEP - крок) наступне виконання програми виконується з зупинками після кожної машинної команди досліджуваної програми.

Перехід на наступну машинну команду .відбувається при натисненні клавіші « [_] ».При цьому в позиціях 1-4 індикується адреса виконуваної

команди, а в позиціях 6-7 - її код. Після кожної зупинки керування передається програмі МОНІТОР. Перед вводом команд необхідно натиснути клавішу « # »що дозволить досліджувати і модифікувати вміст регістрів і пам'яті, що цікавлять, анулювати режим трасування або зупинити виконання програми й оцінити одержані проміжні результати. Потім, якщо необхідно продовжити виконання програми користувачем, вводять команду G без параметрів або ST - кроковий режим.

Слід урахувати, що при покомандному виконанні програми відбувається суттєве уповільнення її виконання.

Приклад 8

Вимагається задати кроковий режим, починаючи з адреси 1A00H. Нижче подано послідовність набору команди

ST 1A0 0 [_] [_] # G SR

у вигляді послідовності натиснення клавіш і видачі повідомлення на індикаторі, як показано в таблиці 8.

Таблиця 8

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
#	-								Встановлення пульта в початковий стан
ST	9	9	9	9					Ввід команди ST

1	9	9	9	1					Ввід адреси
A	9	9	1	A					
0	9	1	A	0					
0	1	A	0	0					
[_]	1	A	0	1		3	E	-	Покроковий режим виконання програми
[_]	1	A	0	2		0	6	-	
#	-								
	9	9	9	9					
CR	-								Команду виконано

4.1.9. Команда Т1

За командою Т1 виконується запуск загальної тест-програми (тесту), до складу якої входять підтести:

-центрального процесора (підтест 1);

-ОЗП (підтест 2);

-РПЗП (підтест 3);

-пристроїв вводу-виводу (підтест 4).

Передбачено можливість одноразового (Т1 CR) і циклічного (ТІ [_]) проходів тест-програми. При натисненні клавіші "Т1" у першій позиції індикатора висвітиться символ « ,» після цього натискують клавіші "CR" або "[_] ".

Після натиснення клавіші "CR" починається послідовне виконання підтестів. На індикоторі при цьому у першій позиції висвітлюється номер підтесту, що виконується. При нормальному завершенні тест-програми у першій позиції висвітиться « ,»

При наявності помилки при виконанні підтесту у першій позиції висвітиться його номер, у позиціях 3-5 - "ЕГГ" (ERROR - помилка). Наприклад, якщо при виконанні першого підтесту (ЦП) відбулася помилка, то на індикаторі буде світитися повідомлення "1 Егг", як показано в таблиці 9.

Таблиця 9

Клавіша	Індикація								Примітки
	1	2	3	4	5	б	7	8	
Т1	,								Ввід тесту Т1

CR	1		Е	Г	Г				Індикація помилки при виконанні перешого підтеста

Якщо відбулася помилка при виконанні другого підтесту (тесту ОЗП) можна процес тестування продовжити наступним чином:
- завершити виконання тесту ОЗП;
- висвітити номер комірки збою та її фактичний та еталоний вміст;
- продовжити виконання тесту ОЗП.

Для того, щоб завершити виконання тесту ОЗП необхідно натиснути клавішу "CR', щоб продовжити виконання тесту ОЗП – клавішу "[_] " ,щоб вивести номер комірки та її вміст - клавішу "F". При натисненні клавіші "F" у перших чотирьох позиціях висвітиться адреса комірки, в наступних двох - вміст комірки, у двох крайніх позиціях - те, що повинно бути записано в комірку за даною адресою.
Наприклад, у комірці за адресою 0075 міститься число 85, а повинно міститися число 13. На індикаторі після завершення тесту ОЗП і натиснення клавіші "F" з'явиться наступна інформація, як показано в таблиці 10.

Таблиця 10

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
	2		Е	Г	Г				
F	0	0	7	5	8	5,	1	3	Індикація інформації після завершення тесту ОЗП і натиснення клавіші "F"

Після того, як на індикаторі висвітиться адреса комірки, в якій відбувся збій, та її вміст, необхідно натиснути клавішу "[_] " або "CR". При неправильному виконанні підтесту 3 (тест РПЗП) на індикаторі висвітиться інформація, як показано в таблиці 11.

Таблиця 11

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
	3		Е	Г	Г		Х	Х	Індикація інформації при не правильному виконанні тесту РПЗП (підтесту 3)

Позначення Х у таблиці 11- це цифри 35, 36 або 37, які вказують номер позиційного позначення ВІС РПЗП, контрольна сума якої виявилась помилковою.

Якщо помилка відбулася при виконанні підтесту 4 (тест пристрою вводу-виводу), у позиціях 7-8 індикатора висвітиться цифра 17,19 або 21: це означає, що помилка відбулася у програмованому інтерфейсі з відповідними позиційними позначеннями ВІС D17, D19 або D21.
При вводі символу "[_]" починається циклічне виконання всіх підтестів.
Вихід із циклічного виконання тест-програми можно здійснити, якщо натискнути клавішу "R" (скид).

4.1.10 Команда Т2

За допомогою команди Т2 здійснюється запуск скороченої тест-програми (тесту 2), призначеної для перевірки КПУ без підключення тесту пристроїв вводу-виводу. Тест 2 складається з наступних підпрограм:
-тест ЦП:
- тест 03П;
-тест РПЗП.
Так само, як і при виконанні команди Т1, передбачено можливість одноразового і циклічного запуску тесту. Виконання камандм Т2 відбувається аналогічно виконанню команди Т1, за вийнятком того, що до складу скороченої тест-програми не входить тест пристроїв вводу-виводу.

4.1.11 Команда ТЗ

За допомогою команди ТЗ здійснюється звертання до тесту 3 КПУ, призначеного для перевірки вводу інформації з пульта та відображення її на індикаторі, як показано в таблиці 12.

Таблиця 12

Клавіша	Індикація								Примітки
	1	2	3	4	5	6	7	8	
ТЗ									Ввід команди ТЗ
CR									Передача керування на виконання
0	0	0	0	0	0	0	0	0	Ввід з пульта символа та відтворення його у всіх 8 позиціях індикатора
1	1	1	1	1	1	1	1	1	
2	2	2	2	2	2	2	2	2	
3	3	3	3	3	3	3	3	3	
4	4	4	4	4	4	4	4	4	
5	5	5	5	5	5	5	5	5	
6	6	6	6	6	6	6	6	6	
7	7	7	7	7	7	7	7	7	
3	8	8	8	8	8	8	8	8	
9	9	9	9	9	9	9	9	9	
A	A	A	A	A	A	A	A	A	
B	B	B	B	B	B	B	B	B	
C	c	C	C	C	C	C	C	c	
D	D	D	D	D	D	D	D	D	
E	E	E	E	E	E	E	E	E	
F	F	F	F	F	F	E	E	F	
#									Встановлення пульта в початковий стан

Перехід від виконання тесту 3 до виконання програш МОНІТОР здійснюють, натиснувши клавішу "R".

5. ВИМОГИ ДО ПРОГРАМНОЇ СУМІСНОСТІ

5.1. При роботі монітора КПУ використовується частина обладнання та оперативної пам'яті контролера. Це визначає ряд обмежень при розробці програми користувача.

При експлуатації контроллера сумісно з пультом:

1) не допускається використовувати 0-канал таймера, оскільки він використовується для виконання команди ST;

2) у програмі не допускається використовувати нулевий і перший вектори переривання, оскільки вони використовуються для організації роботи програми МОНІТОР;

3) вміст комірки за адресою 2000Н програми користувача повинно дорівнювати СЗН;

4) точка входу в програму користувача повинна мати адресу 5020Н, глибина стека, що використовується, - не більше ЗО байті в;

5)вміст робочих комірок програми МОНІТОР (оперативної пам’яті 1800Н—1899Н) не повинен змінюватися програмами користувача.

Примітка. Якщо вміст комірки 2000Н не дорівнює СЗН, то автоматичного запуску програми користувача не відбувається. В цьому випадку для запуска програми користувача необхідно з пульта КПУ ввести команду 6 і початкову адресу програми (наприклад, G 1920 ("CR").

5.2. Можливі два варіанти обробки переривань.

Перший варіант: контролер переривань КР1810ВН9А працює в режимі, який заданий програмою ініціалізації МОНІТОРА. В цьому випадку передбачені наступні вектори переривань, які показані в таблиці.

Таблиця 13

Рівень переривання	Вектор переривань для контролера "Електроніка МС2721"
2	2000Н
3	2004Н
4	2008Н

5	200CH
6	2010H
7	2014H

Користувач за вказаними адресами повинен записати команди переходів на підпрограми обробки відповідних рівнів переривань. Інший варіант: користувач програмує контролер - переривань КР1810ВН9А, встановлюючи зручні для нього вектори переривань. В цьому випадку в підпрограмі обробки переривання за нульовим рівнем повинен бути записаний перехід на адресу 100Н, тобто ЛМР 100Н, в підпрограмі обробки переривання за першим рівнем - перехід на адресу 104Н, тобто ЛМР 104Н.

6. ПОВІДОМЛЕННЯ

Нижче подано список помилок, які можуть виникнути в процесі діалогової взаємодії користувача і КПУ для контролера "Електроніка МС2721":

ЕГГ 1 - не існує такої команди;
ЕГГ 2 - не введено команду;
ЕГГ 3 - не відповідає число введених параметрів формату команди;
ЕГГ 4 - виконання команди закінчується натисненням клавіші "СІ?";
ЕГГ 5 - введені неправильні параметри для команди М;
ЕГГ 6 - помилкова клавіша.
Скинути помилковий стан можно тільки натисненням клавіші « # »

7. ПОРЯДОК ВИКОНАННЯ РОБОТИ

Завдання для підготовки до роботи в лабораторії (самопідготовка)

Вивчити склад і основні функції КПУ "Електроніка МС 2721".

Накреслити блок-схему КПУ.

Підготуватизаготовку звіту про лабораторну роботу.

Заготувати таблицю результатів роботи за пунктами 7.4 - 7.8.

Завдання для виконання роботи

7.4. Після ввімкнення живлення послідовно натиснути клавіш "R" і "#". У крайній лівій позиції індикатора (позиції 1) символ "_"

свідчить про готовність контролера до роботи. Потім вводиться ім'я команди. Якщо команда потребує вводу параметрів, то в крайній правій позиції індикатора висвітлюються цифри (послідовно) 1;2... Параметри, що вводяться, розділяються знаком пробілу. Натискання клавіші "CR" дає сигнал на виконання команди. Вийняток: команди S, X - виконуються також після натиснення клавіші (пробіл).
Порядок набору і результати виконання кожної команди заносяться до звіту.

7.5. Провести тестування контролера, використавши команди МОНІТОРА Т2 і Т3.

7.5.1 Провести тестування ЦП, ОЗП, РПЗП за допомогою команди Т2 CR.

7.5.2 Провести тестування пульта керування за допомогою команди Т3 CR, послідовно набираючи всі символи з клавіатури пульта.

7.6. Дослідження використання команди S. Для цього:

7.6.1 За допомогою команди S заповнити числом (байтом) В3Н:
- першу область пам’яті, наприклад, з адресами 18A0H—18AFH;
- другу область пам’яті, наприклад, з адресами 18B0H—18BFH

Виконання:
S початкова адреса [_] байт [_] ... [_] байт CR
області пам'яті даних даних
S 18A0 [_] B3 [_] ... [_] B3 CR
S 18B0 [_] B3 [_] ... [_] B3 CR

7.6.2. Використовуючи команду S, проконтролювати вміст обох областей пам'яті (наприклад, з адресами I8A0H-18AFH; 18B0H-8BFH).
Виконання:
S початкова адреса [_] індикація [_] ... [_] індикація
контрольованої вмісту вмісту
області пам'яті
S 18A0 [_] [_]...[_]
S 18B0 [_] [_]...[_]

7.6.3 За допомогою команди S модифікувати вміст комірок, наприклад з адресами 18B0H-18BFH, записавши за ними нові дані, наприклад, B1H.
Виконання:
S адреса [_] індикація вмісту [_]...[_] індикація вмісту
комірки комірки пам'яті комірки пам’яті
пам'яті та його модифікація та його модифікація
S 18B0 [_] B1 [_] ... [_] B1 CR

ПРАКТИЧНА РОБОТА 4

ВИВЧЕННЯ КОМАНД ПЕРЕДАЧІ ДАНИХ МІКРОПРОЦЕСОРА СЕРІЇ І8080

Мета роботи:

1. Вивчення команд передачі даних.
2.Отримання навичок створення простих програм на мові АСЕМБЛЕРА.
3.Освоєння порядку асемблювання програм і вводу робочої програми в ОЗП.
4.Освоєння процесу налагодження простої програми.

1. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Починаючи вивчати команди цієї групи, слід пам'ятати, що вони реалізують операції обміну даними між регістрами і пам’яттю, а також між акумулятором і пристроями вводу-виводу.
Зміст регістра ознак при виконананні цих команд не змінюється.
Порядок запису джерела і приймача даних у командах визначається типом мікропроцесора. Для мікропроцесора серії КР580 спочатку вказується приймач, потім джерело даних.

Команда MOV

Важливо враховувати, що ця команда служить для передачі даних між регістрами і комірками пам’яті, які адресовані непрямим чином.
Довжина команди MOV 1 байт. У полі операндів вказуються або імена регістрів, або ім’я регістра і символічне ім'я М, котре визначає комірку пам'яті (адресу якої утримує регістрова пара HL).

Приклад 1
Передати зміст регістра В в акумулятор.
Розв'язок: MOV A,B

Приклад 2
Передати зміст комірки пам'яті, адреса якої записана в регістровій парі Hl - в регістр D.
Розв’язок: MOV D,M

Приклад 3
Передати вміст регістра С в комірку пам’яті, адреса якої вказана в регістровій парі HI.
Розв'язок: MOV M,C

Команда MVI

Необхідно розуміти відмінність цієї команди від команди МОУ: тут джерелом даних є 8-бітна константа, тобто використовується безпосереднє адресування. Приймачем даних може бути регістр, або комірка пам’яті, адресована непрямим чином. Довжина команди 2 байти.

Приклад 4
Записати двійкову константу 01011110B у комірку пам'яті, адреса якої міститься в регістровій парі Hl.
Розв’язок: MVI M,01011110B.
Літера B після коду числа позначає використання двійкової системи числення.

Приклад 5
Записати шістнадцяткову константу 3AH в регістр D.
Розв’язок: MVI D,3AH.
Літера H в кінці коду числа позначає використання шіснадцяткової системи числення.

Команда LXI

Ця команда застосовується для завантаження шістнадцяткового числа (адреси комірки пам’яті) в одну з регістрових пар BC, D E або H L. Вміст другого байта передається в молодший регістр регістрової пари, а вміст третього - у старший.

Приклад 6
Записати в регістрову пару D E шістнадцяткове число 3A8FH.
Розв'язок 1: LXI D,3A8FH
Розв'язок 2: MVI D,3AH MVI E, 8FH
Розв’язок 1 кращий, оскільки для запису команди використовується 3 байти, а у розв'язку 2 кожна команда займає 2 байти, всього треба 4 байти.

Команди LDA, STA

Ці команди використовуються для обміну даними між акумулятором і пам’яттю. Адреса комірки пам’яті вказується явно: у другому байті

команди розміщується молодший байт адреси пам’яті, а у третьому - старший.
LDA - передача даних із пам’яті в акумулятор.
STA - передача даних із акумулятора в пам'ять.

Приклад 7
Передати наявність акумулятора в комірку пам’яті з адресою 2FCAH.
Розв'язок: STA 2FCAH

Команди LDAX, STAX

Ці команди за призначенням аналогічні командам LDA, STA, але комірка пам'яті адресується непрямим чином в регістрових парах BC або DE. В команді вказується тільки ім'я старшого регістра (B або D). Довжина команди 1 байт.

Приклад 8
Передати в акумулятор вміст комірки пам’яті, адреса якої вказана в регістровій парі DE.
Розв'язок: LDAX D

Команди LHLD, SHLD

Ці команди існують для обміну даними між регістровою парою HL і пам’яттю. Адреса комірки пам’яті вказується в другому і третьому байтах команди (пряма адресація). У цю комірку передається вміст регістра L (за командою SHLD) або вміст цієї комірки передається в регістр L (за командою LHLD). При цьому регістр H обмінюється даними з коміркою пам'яті з наступним номером.

Приклад 9
Передати вміст регістра L в комірку пам’яті з адресою 2AF4H, а вміст регістра H - в комірку з адресою 2AF5H.
Розв’язок: SHLD 2AF4H

Команда XCHG

Ця команда виконує взаємний обмін даними між регістрами H, L і D,E Адресування непряме. Довжина команди 1 байт.
Приклад 10
Обміняти вміст регістрових пар H,L і D,E.
Розв'язок: XCHG

Команди IN, OUT

Ці команди забезпечують-обмін даними між акумулятором і пристроями вводу-виводу (ПВВ). У другому байті вказується номер порту вводу (в команді IN) або виводу (в команді OUT). Довжина команди 2 байти.

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ

Завдання для підготовки до роботи в лабораторії (самопідготовка)

2.1. Вивчити систему команд мікропроцесора серії КР580. Засвоїти призначення основних груп команд.

2.2. Вивчити склад, призначення і порядок застосування команд передачі даних.

2.3. Скласти просту програму на мові асемблера з застосуванням команд передачі даних (за вказікою викладача).

2.4. Провести процес ручного асемблювання складеної програми.

2.5. Записати одержану після асемблювання програму у вигляді таблиці адрес і даних, підготувавши її таким чином для завантаження в ОЗП.

2.6. Підготувати заготовку звіту про лабораторну роботу з відображенням результатів за пунктами 2.1-2.5. Заготувати таблицю результатів роботи в лабораторії згідно пунктів завдання.

Завдання для виконання роботи

2.7. Увімкнути живлення контролера. Встановити вихідний стан, провести тестування.

Подальше вивчення команд мікропроцесора виконується в такому порядку:

2.8. Відповідно до завдання, вказаного для вивчення команди, складаються команди на мові асемблера.

2.9. Виконується ручне асемблювання складених команд.

2.10. За допомогою команд монітора:

- завантажуються дані у пам’ять і регістри мікропроцесора:
- одержані машинні команди і дані завантажуються в ОЗП:
- здійснюється виконання команд та програм:
- виконується перевірка виконання команд та програм.

2.11. Дослідити застосування команд передачі даних:

MOV, MVI, LXI, LDA, STA, LDAX, STAX, LHLD,. SHLD, XCHG.

Порядок роботи згідно кожного пункту завдання і результати виконання кожної команди занести в таблицю результатів.

Примітка : у наступних завданнях адреси та виконання програм надані з урахуванням ОЗП та команд монітора для контролера програмованого універсального "Електроніка МС2721".

Застосування команди MOV

Завдання 1

Передати вміст регістра В в акумулятор (попередньо занести в регістр В число, наприклад, 37Н командою монітора).

Програму подано у вигляді таблиці 1.

Таблиця 1

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
O R G 1 9 0 H	АДРЕсА	ДАНІ	
M0V A,B	1901	78	Передати вміст регістра B в акумулятор (регістр A)
H LT	1902	76	Зупинка програми

Виконання:

```
#$ 1901 [_] 78 [_] 76 CR        ;Занесення програми в ОЗП.
#X [_] 37 CR                    :Занесення числа 37Н в регістр В.
#G 1901 [_] 1902 CR             ;Запуск програми на виконання.
#X                              ;Індикація (контроль) вмісту регістра А.
```

Результат виконання програми занести у звіт.

Завдання 2

Передати вміст регістра С в комірку пам’яті, адреса якої вказана в регістровій парі HL (попередньо занести в С число, наприклад, ЗАН, в HL- адресу, наприклад, 1900Н).

Програму подано у вигляді таблиці 2.

Таблиця 2

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
O RG 1910H	АДРЕСА	ДАНІ	
MOV M,C	1910	71	Передати вміст регістра С в комірку пам'яті
H LT	1911	76	Зупинка програми

Виконання:
#S1910 [_] 71 [_] 76 CR ;Занесення програми в 03П.
#X [_] [_] 3A [_] [_] [_] 19 [_] 00 CR ;Занесення даних в регістри C,H,L
#G 1910 [_] 1911 CR ;Запуск програми на виконання.
#S 1900 [_] ;Індикація (контроль) вмісту комірки з адресою 1900H

Результат виконання програми занести у звіт

Застосування команди MVI

Завдання 3
Записати шістнадцяткову константу ЗАН в комірку пам'яті, адреса якої міститься в регістрі H,L (адреса 1900H).
Програму подано у вигляді таблиці 3.

Таблиця 3 -

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
ORG 1920H	АДРЕСА	ДАНІ	
MVI M, ЗАН	1920 1921	36 3A	Передати константу ЗАН в комірку пам'яті
HLT	1922	76	Зупинка програми

Виконання:
#S 1920 [_] 36 [_] 3A [_] 76 CR ;Занесення програми в ОЗП.
#X [_] [_] 3A [_] [_] [_] 19 [_] 00 CR ;Занесення даних у регістриH,L
#G 1920 [_] 1922 CR ;Запуск програми на виконання.
#S 1900 [_] ;Індикація (контроль) вмісту комірки з адресою 1900H.

Результат виконання програми занести у звіт.

Застосування команди LXI

Завдання 4
Записати в регістрову пару D,E шістнадцяткове число, наприклад 1900H.
Програму подано у вигляді таблиці 4.
Таблиця 4

АСЕМБЛЕР	Об'єктні коди		КОМЕНТАР
ORG 1940H	АДРЕСА	ДАНІ	
LXI D,1900H	1940 1941 1942	**11** 00 19	Передати константу 1900H в регістрову пару DE
HLT	1943	76	Зупинка програми

Виконання:
#S 1940 [_] 11 [_] 00 [_] 19 [_] 76 CR ;Занесення програми в ОЗП.
#G 1940 [_] 1943 CR ;Запуск програми на виконання.
#X [_] [_] [_] індикація [_] індикація ; Індикація, (контроль) вмісту :регістрів D, E.

Результат виконання програми занести у звіт.

Застосування команд LDA, STA

Завдання 5

Передати вміст акумулятора в комірку пам’яті з адресою 193АН (попередньо занести в А число, наприклад, В5Н).

Програму подано у вигляді таблиці 5.

Таблиця 5

АСЕМБЛЕР	ОБ’ЄКТНІ КОДИ		КОМЕНТАР
ORG 1950	АДРЕСА	ДАНІ	
STA 193АН	1950	32	Передати вміст акумулятора
	1951	3А	в комірку пам'яті
	1952	19	з адресою 193АН
HLT	1953	76	Зупинка програми

Виконання:

#S 1950 [_] 32 [_] 3A [_] 19 [_] 76 CR ;Занесення програми в ОЗП.

#X B5 CR :Занесення числа В5Н в акумулятор.

#G 1950 [_]1953 CR :Запуск програми на виконання

#S 193АН [_] :Індикація (контроль) вмісту ;комірки з адресою 193АН.

Результат виконання програми занести у звіт.

Завдання 6

Передати вміст комірки пам’яті (наприклад, 5АН) з адресою 193АН в акумулятор.

Скласти програму, асемблювати і виконати самостійно.

Розв'язок записати у вигляді таблиці.

При підготовці до виконання програми необхідно передбачити:

- занесення програми в ОЗП :
- занесення числа (наприклад, 5АН в комірку пам'яті ОЗП за адресою I93АН:
- запуск програми:
- контроль результату виконання.

Результат виконання програми занести у звіт.

Застосування команд LDAX, STAX

Завдання 7

Передати вміст комірки пам’яті (наприклад, 47H) з адресою 1940H в акумулятор із застосуванням команди LDAX.

Скласти програму, асемблювати і виконати самостійно.

Розв’язок записати у вигляді таблиці.

При підготовці до виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення числа (наприклад, 47H) в комірку пам’яті ОЗП за адресою 1940H:
- запуск програми:
- контроль результату виконання.

Результат виконання програми занести у звіт.

Завдання 8

Передати вміст акумулятора (наприклад, ЗАН) в комірку пам'яті ОЗП з адресою 1950H із застосуванням команди STAX.

Скласти програму, асемблювати і виконати самостійно.

Розв'язок записати у вигляді таблиці.

При підготовці до виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення числа (наприклад, ЗАН) в акумулятор;
- запуск програми;
- контроль результату виконання.

Результат виконання програми занести у звіт.

Застосування команд LHLD, SHLD

Завдання 9

Передати вміст регістрової пари H,L (наприклад, I950H) в комірку пам'яті ОЗП з адресою 1970H.

Скласти програму, асемблювати і виконати самостійно.

Розв’язок записати у вигляді таблиці.

При підготовці до виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення числа (наприклад, 1950H) в регістрову пару H,L;
- запуск програми;
- контроль результату виконання.

Результат виконання програми занести у звіт.

Завдання 10
Передати вміст комірок пам'яті ОЗП з адресами 1980Н, 1981Н (наприклад, 5АН та 6ВН відповідно) в регістрову пару H,L.
Скласти програму, асемблювати і виконати самостійно.
Розв'язок записати у вигляді таблиці.
При підготовці до виконання програми необхідно передбачити:
- занесення програми в ОЗП ;
- занесення чисел (наприклад, 5АН та 6ВН) в комі роки пам'яті ОЗП з адресами 1980Н, 1981Н;
- запуск програми;
- контроль результату виконання.
Результат виконання програми занести у звіт.

Застосування команди XCHG

Завдання 11
Передати вміст регістрової пари H,L (наприклад, 1950Н) - в регістрову пару D,E.
Скласти програму, асемблювати і виконати самостійно.
Розв'язок записати у вигляді таблиці.
При підготовці до виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення числа (наприклад, 1950Н) в регістрову пару H,L;

запуск програми;

- контроль результату виконання.

Результат виконання програми занести у звіт.

ПРАКТИЧНА РОБОТА 5

ВИВЧЕННЯ АРИФМЕТИЧНИХ КОМАНД МІКРОПРОЦЕСОРА СЕРІЇ I8080

Мета роботи:
1. Вивчення команд арифметичних команд мікропроцесора серії I8080
2.Отримання навиків створення простак програм на мові АСЕМБЛЕРА мікропроцесора серії I8080 з використанням арифметичних команд.

1. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Приступаючи до вивчення цієї групи, слід знати, що вони забезпечують виконання арифметичних операцій. У мікропроцесорі

серії I8080 це операції додавання і віднімання. При цьому один з операндів повинен бути раніше . занесений в акумулятор (непряме адресування). Адреса другого операнда вказується у команді. Спосіб його адресування визначає довжину команди. Результат операції помішується в акумулятор, знищуючи операнд, який був у ньому раніше.

Всі команди цієї групи змінюють ознаки стану регістра ознак (регістра F): "нуль’’, "знак", ’’перенос", "напівперенос", "паритет".

При відніманні виконується додавання зменшуваного з там, що віднімається, представленим у додатковому коді. При цьому ознака "перенос" відповідає ознаці "займу".

Команди ADD, SUB

Важливо знати, що ці команди використовуються для виконання додавання (ADD) або віднімання (SUB), коли другий операнд знаходиться в регістрі загального призначення або пам'яті. У першому випадку в команді після позначення операції вказується ім'я регістра, в другому - символічне ім'я. "M" (MEMORY). Останнє означає, що операнд вилучається з комірки пам'яті, адресу якої містить регістрова пара HL (непряма операція). Довжина команд ADD і SUB 1 байт.

Приклад 1

Додати число з комірки пам'яті з адресою 11F7H до числа з комірки пам'яті з адресою A728H. Результат помістити на місце другого операнда.

```
Розв'язок:   LDA 11F7H ;Вміст комірки пам’яті з адресою 11F7H передається в акумулятор.
             LXI H,A728H :Адреса другого доданку завантажується ;в регістрову пару HL.
             ADD M ;Число, яке записане в акумуляторі, складається з числом, яке знаходиться в ;комірці пам'яті з адресою A728H.
             MOV M,A ;Вміст акумулятора (сума) передається в ту саму комірку.
```

Команди ADI, SUI

Важливо засвоїти, що ці команди відрізняються від розглянутих вище команд ADD,SUB вказуванням безпосереднього операнда, який записується в другому байті команди у вигляді двійкової, десяткової або шістнадцяткової константи.
Збільшити вміст акумулятора на величину 00010000B.
Розв'язок: ADI 00010000B.

Команди ADC, SBB

Ці команди відрізняються тим, що в заданій операції діють не тільки два операнди, як в командах ADD, SUB, але й ознака переносу, яка залишилася від попередньої операції. При виконанні команди ADC значення ознаки переносу додається до суми опєрандів. При виконанні команди SBB додатково з різниці двох опєрандів віднімається значення ознаки переносу (займа). Довжина цих команд 1 байт. Ці команди використовуються при обробці опєрандів довжиною-понад один байт.

Команди ACI, SBI

Ці команди забезпечують виконання додавання і віднімання з урахуванням ознаки переносу, як і команди ADC, SBB, але другий операнд вказується в них безпосередньо. Таким чином довжина цих команд - 2 байти.

Команди INR, DCR, I NX, DCX

До групи команд арифметичних операцій відносять команди збільшення або зменшення на 1 вмісту:
- регістра:
  INR ім'я регістра:
  DCR ім’я регістра;
- комірки пам'яті, адресованої непрямим чином за допомогою регістрової пари HL:
  INR M;
  DCR м;
-регістрової пари:
  I NX ім'я старшого регістра регістрової пари (B, D або H):
  DCX ім'я старшого регістра регістрової пари (B, D або H)

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ

Завдання для підготовки до роботи в лабораторії (самопідготовка)

1.1. Вивчити склад, призначення і порядок . застосування команд арифметичних операцій.

1.2. Скласти просту програму на мові асемблера із застосуванням команд арифметичних операцій (за вказікою викладача).

1.3. Провести процес ручного асемблювання складеної програми.

1.4. Записати одержану після асемблювання програму у вигляді таблиці адрес і даних, підготувавши її таким чином для завантаження в ОЗП.

1.5. Підготувати заготовку звіту про лабораторну роботу з відображенням результатів за пунктами 2.1-2.4. Заготувати таблицю результатів роботи в лабораторії по пунктах завдання.

Завдання для виконання роботи

1.6. Увімкнути живлення контролера. Встановити вихідний стан, провести тестування.

Подальше вивчення команд мікропроцесора виконується в такому порядку:

1.7. Відповідно до завдання, вказаного для вивчення команди, складаються команди на мові асемблера.

1.8. Виконується ручне асемблювання складених команд.

1.9. За допомогою команд монітора:

- завантажуються дані в пам’ять і регістри мікропроцесора:
- одержані машинні команди і дані завантажуються в ОЗП:
- здійснюється виконання команд та програм:
- виконується перевірка виконання команд та програм.

1.10. Дослідити застосування команд арифметичних операцій:

ADD, SUB, ADI, SUI, ADC, SBB, ACI, SBI.

Порядок роботи згідно кожного пункту завдання і результати виконання кожної команди занести в таблицю результатів.

Примітка: у наступних завданнях адреси та виконання програм надані з урахуванням ОЗП та команд монітора для контролера програмованого універсального "Електроніка МС2721”.

Застосування команд ADD, SUB

Завдання 1

Скласти число з регістра D (наприклад, число EBH) з числом, яке зберігається в акумуляторі (наприклад, число 93H).

Програму подано у вигляді таблиці 1.

Таблиця 1

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
0RG 1960H	АДРЕСА	ДАНІ	
ADD D	1960	82	Складання вмісту регістра D з вмістом акумулятора
HLT	1961	76	Зупинка програми

Асемблювати і виконати програму самостійно.

При підготовці до виконання програми необхідно передбачити:

- занесення програми в ОЗП :
- занесення чисел (наприклад, EBH та 93H) в регістр D та акумулятор відповідно:
- запуск програми:
- контроль результату виконання.

Результат виконання програми занести у звіт.

Завдання 2

Додати число з комірки пам'яті з адресою 19A0H (наприклад, число A5H) до числа з комірки пам'яті з адресою 19A2H (наприклад, число 37H). Програму подано у вигляді таблиці 2.

Таблиця 2

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
0RG 1970H	АДРЕСА	ДАНІ	
IDA 19A0H	1970	3A	Завантажити акумулятор із
	1971	A0	комірки 19A0H
	1972	19	

LXI H,19A2H -	1973 1974 1975	21 A2 19	Завантажити в регістрову пару HL адресу 19A2H
ADD M	1976	86	Скласти вміст акумулятора з вмістом комірки пам'яті M
LXI H,19A5H	1977 1978 1979	21 A5 19	Завантажити в регістрову пару HL адресу 19A5H
MOV M,A	197A	77	Передати вміст акумулятора в пам'ять
HLT	197B	76	Зупинка програми

Асемблювань і виконати програму самостійно.
При підготовці до виконання програми необхідно передбачити:

- занесення програми в ОЗП ;

- занесення чисел (наприклад, A5H та 37H) в комірки пам'яті з адресами 19A0H та 19A2H відповідно;
- запуск програми;
- контроль результату виконання.

Результат виконання програми занести у звіт.
Завдання 3
Відняти від числа, яке зберігається в комірці пам’яті з адресою 19A0H (наприклад, число EBH) число, яке міститься в регістрі B (наприклад, число 75H). Результат залишити в акумуляторі.
‘ Скласти програму, асемблювати і виконати самостійно.
Розв'язок записати у вигляді таблиці.
При підготовці до виконання програми необхідно передбачити:
- занесення програми в ОЗП ;
- занесення числа (наприклад, EBH) в комірки пам'яті з адресами 19A0H;
- занесення числа (наприклад, 75H) в регістр B;
- запуск програми;
- контроль результату виконання.

Результат виконання програми занести у звіт.

Застосування команд ADC, SBB

Завдання 4
Скласти 2 двобайтних числа, які зберігаються в регістрах B, C (наприклад, число EA5AH) і- регістрах D, E (наприклад, число 6B73H). Результат занести в регістри H, L.
Програму подано у вигляді таблиці 3.
Таблиця 3

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
0RG 1980H	АДРЕСА	ДАНІ	
MOV A,C	1980	79	Передати вміст регістра С в акумулятор (регістр А)
ADD E	1981	83	Додавання до вмісту регістра А вміст регістра Е (додавання молодших байтів чисел)
MOV L,A f	1982	4F	Передати вміст акумулятора в регістр L (запам'ятати молодший байт результату додавання молодших байтів чисел)
MOV A,B	1983	78	Передати вміст регістра В в акумулятор
ADC D	1984	8A	Додавання до вмісту акумулятора вміст регістра 0 з урахуванням ознаки переносу (додавання старших байтів чисел з урахуванням ознаки переносу)
MOV H,A	1985	47	Передати вміст акумулятора в регістр H (запам'ятати старший байт результату додавання старших байтів чисел)
HLT	1986	76	Зупинка програми

Асемблювати і виконати програму самостійно.
Розв'язок записати у вигляді таблиці.
При підготовці до виконання програми необхідно передбачити:

- занесення програми в ОЗП ;

- занесення чисел (наприклад, EA5AH та 6B73H) в регістри B, C та 0, E відповідно;
- запуск програми;
- контроль результату виконання.

Результат виконання програми занести у звіт.

Завдання 5

Відняти двобайтні числа. Наприклад, від числа EA5AH відняти 6B73H. Перше число записане в адресах 1A75H, 1A76H, друге - 1A79H, 1A7AH (відповідно молодші і старші байти).

Скласти, асемблювати і виконати програму самостійно.

Розв'язок записати у вигляді таблиці.

При підготовці до виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення чисел (наприклад, EA5AH та 6B73H) в комірки пам'яті за адресами 1A75H, 1A76H, 1A79H, 1A7AHE відповідно;
- запуск програми;
- контроль результату виконання.

Результат виконання програми занести у звіт.

Використання команд ADI, SUI

Завдання 6

Збільшити вміст акумулятора на величину ЗАН (наприклад, в акумуляторі міститься число 49H).

Програму подано у вигляді таблиці 4.

Таблиця 4

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
ORG 1990	АДРЕСА	ДАНІ	
ADI ЗАН	1990 1991	C6 ЗА	Складання числа ЗАН з вмістом акумулятора
HLT	1992	76	Зупинка програми

Асемблювата і виконати програму самостійно.
Розв'язок записати у вигляді таблиці.
При підготовці до виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення числа (наприклад, 49Н) в акумулятор;
- запуск програми;
- контроль результату виконання (індикація вмісту, регістра А).

Результат виконання програми занести у звіт.

Завдання 7

Відняти від вмісту акумулятора (наприклад, в акумуляторі міститься число 54Н) число ЗВН.
Скласти, асемблювати і виконати програму самостійно.
Розв'язок записати у вигляді таблиці.
При підготовці'до виконання програми необхідно передбачити:
- занесення програми в ОЗП;
- занесення чисел (наприклад, ЕА5АН та 6В73Н) в регістри В, С та 0, Е відповідно;
- запуск програми;
- контроль результату виконання.
Результат виконання програми занести у звіт.

Використання команд ACI, SBI, INX

Завдання 8

Скласти 2 двобайтних числа, від результату відняти двобайтне число. Перше число міститься в комірках пам'яті за адресами 1970Н, 1971Н, друге - за адресами 1972Н, 1973Н. Наприклад до числа ЕВ73Н додати число 6А85Н, від результата відняти число 0110Н. Результат помістити в комірки з адресами 1974Н, 1975Н (молодший і старший байти відповідно).
Програму подано у вигляді таблиці 5.
Асемблювати і виконати програму самостійно.
Розв'язок записати у вигляді таблиці.
При підготовці до виконання програми необхідно передбачити:
- занесення програми в ОЗП.;
- занесення чисел наприклад, ЕВ73Н та 6А85Н в комірки пам’яті за адресами 1970Н, 1971Н, 1972Н, 1973Н відповідно;
- запуск програми;
- контроль результату виконання.
Результат виконання програми занести у звіт.

Таблиця 5

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
	АДРЕСА	ДАНІ	
LXI H,1970H			Завантаження адреси старшого байта першого доданка в регістри H,L
MOV B,M			Передача старшого байта першого доданка в регістр B
I NX H			Обчислення адреси молодшого байта першого доданка
MOV C,M			Передача молодшого байта першого доданка в регістр C
I NX H			Обчислення адреси старшого байта другого доданка
MOV D,M			Передача старшого байта другого доданка в регістр D
I NX H			Обчислення адреси молодшого байта другого доданка
MOV A,M			Передача молодшого байта другого доданка в акумулятор
ADD C			Складання молодших байтів двох доданків
MOV C,A			Запис результату суми в регістр C
I NX H			Обчислення адреси старшого байта другого доданка
MOV A,M			Передача старшого байта другого доданка в акумулятор

ADC B			Складання старших байтів двох доданків з врахуванням переносу
MOV B,A			Запис результату суми в регістр B
MOV A,C			Передача молодшого байта суми в акумулятор
SUI 1OH			Віднімання молодшого байта константи
STA 1975H			Запис результату в комірку пам’яті з адресою 1975H
MOV A,B			Передача старшого байта суми в акумулятор
SBI 01H			Віднімання старшого байта константи з урахуванням займу
STA 1974H			Запис старшого байта результату в комірку пам’яті з адресою 1974H
HLT			Зупинка програми

ПРАКТИЧНА РОБОТА 6

ВИВЧЕННЯ ЛОГІЧНИХ КОМАНД МІКРОПРОЦЕСОРА СЕРІЇ КР580

Мета роботи:
1. Вивчити склад, призначення та застосування логічних команд мікропроцесора серії КР580.
2. Отримати навички складання простих програм з використанням логічних команд.

1. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Команди логічних операцій мікропроцесора серії I8080
Слід знати, що команди цієї групи забезпечують логічні операції.
Мікропроцесор серії КР580 має такі логічні операції:

- логічне множення ("і");
- логічне додавання ("АБО"):

- нерівнозначність ("ВИКЛЮЧАЮЧЕ АБО"):
- порівняння двох чисел:
- зсув вмісту акумулятора праворуч або ліворуч;
- інвертування вмісту акумулятора:
- встановлення та інвертування ознаки переносу.

При цьому необхідно пам'ятати, що логічні операції є порозрядними, тобто виконуються незалежно для всіх 8 біт операндів. Один з операндів знаходиться в акумуляторі (неявна адресація), туди ж завантажується і результат операції.

В таблиці і представлено значення результатам в виконання логічних операцій "І" (У=Х1*Х2), "АБО" (Z=X1*X2), "ВИКЛЮЧАЮЧЕ АБО" (Р=Х1(+)Х2) для двох однобітних операндів Х1, Х2.

Таблиця і - Логічні операції "І[й]", "АБО", "ВИКЛЮЧАЮЧЕ АБО"

XI	X2	Y=X1*X2	Z=Xl*X2	P=X1(+)X2
0	0	0,	0	0
0	1	0	1	1
1	0	0	1	1
1	1	*4* 1	1	0

При виконанні команд, відповідних цим операціям, формуються всі ознаки, крім ознак переносу і допоміжного переносу, які встановлюються в нуль.

У регістрі ознак використовуються тільки п'ять розрядів.

Розряд 0 - розряд переносу С (CARRY). В нього записується 1, якщо при виконанні арифметичної команди або команди зсуву було переповнення акумулятора. В протилежному випадку в цей розряд записується 0.

Розряд 2 - розряд парності Р (PARITY). В нього записується 1, якщо при виконанні команди кількість одиниць в розрядах акумулятора буде парна.

Розряд 4 - допоміжний розряд переносу (міжтетрадний перенос) АС (AUX CARRY). В нього записується 1, якщо при виконанні команд в акумуляторі виникає одиниця переносу з третього розряду числа.

Розряд 6 - розряд ознаки нуля Z (ZERO). В нього записується 1, якщо при виконанні арифметичної або логічної команди у всіх розрядах числа в акумуляторі є 0, в протилежному випадку в розряд записується 0.

Розряд 7 - розряд знака S (SIGN). В нього записується 1, якщо при виконанні арифметичної команди отримане число від'ємне. В протилежному випадку в цей розряд записується 0.
Формат байта регістра ознак показано на рис.1.

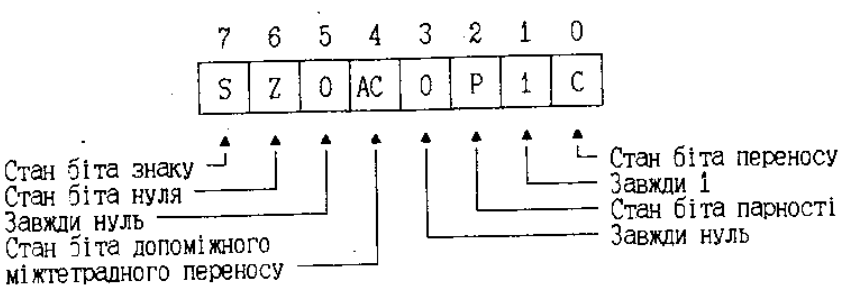


Рис.1. Формат байта регістра ознак

У більшості випадків при виконанні програм необхідно перевіряти або міняти (маскувати) стан одного або декількох розрядів числа в акумуляторі. Це можна виконати за допомогою таких операцій*

1) логічне множення числа в акумуляторі і маски, яка очищує розряд числа, якщо у відповідному розряді маски буде записано 0, і незмінює його, якщо в розряді маски записано 1;
2) логічне сумування числа в акумуляторі і маски, яка встановлює розряд числа в 1, якщо в такому ж розряді маски буде записано 1, і незміне його, якщо в цьому розряді записано 0;
3) логічне "виключаюче АБО" числа в акумулятор і числа яке інвертує наявність розряду числа, якщо у відповідному розряді маски записано 1, і незмінює його, якщо в цьому розряді записано 0.

Команди логічного множення ANA, ANI

Необхідно знати, що ці команди виконують порозрядне логічне множення операндів. Для вказування другого операнда в команді ANA використовується регістрова або непряма, адресація. Довжина цієї команди 1 байт.
У команді ANI другий операнд вказується беспосередньо в другому байті. При цьому слід пам’ятати, що операція логічного множення використовується для перевірки значення певного біта числа в акумуляторі за допомогою другого числа-маски. Наприклад, якщо

потрібно перевірити стан третього біта числа в акумуляторі, то маска повинна мати вид 00001000B. Після виконання команди логічного множення в регістрі прапорців (регістр F) за ознакою нуля (Z) можна говорити про стан третього біта числа.
За допомогою цієї ж команди можна, використовуючи число-маску, скидати певні біта числа в акумуляторі. Для цього у відповідних розрядах повинні бути нулі.
Приклад 1
Виконати операцію логічного множення над вмістом комірок пам'яті 1А32Н і 1А33Н. Вміст п'ятого біта результату занести в комірку 1А34Н.
Розв'язок: LXI H, 1A32H: Запис адреси 1A32H в регістрову пару H,L.
MOV A,M: Передача першого операнда з комірки з адресою 1A32H вакумулятор.
I NX H: Збільшення вмісту регістрової пари H,Lна 1 (обчислення адреси наступного числа).
ANA M: Виконання логічного множення числа, записаного в акумуляторі і числа з комірки 1А33Н.
ANI 00100000B I NX H: Виділення п'ятого біта результату.
INX H Збільшення вмісту регістрової пари H,L на 1 (обчислення адреси результату).
M0V M,A: Запис вмісту акумулятора в комірку пам'яті з адресою 1A34H.
H LT:Зупинка

Команди ORA, ORI

Ці команди виконують порозрядно логічне додавання операндів. Адресація операндів і довжина такі самі, як у командах ANA, ANI. Ці команди використовуються для установки визначених бітів числа в акумуляторі в одиницю, а також для об'єднання окремих бітів різних чисел (слів) у нове слово.
Приклад 2
Встановити старші біти числа в акумуляторі в 1.
Розв'язок: ORI 11110000B

Команди XRA, XRI

Необхідно знати, що ці команди виконують поразрядно операцію нерівнозначності С ВИКЛЮЧАЮЧЕ АБО, ДОДАВАННЯ ЗА

МОДУЛЕМ 2). Адресація і довжина аналогічні відповідно командам ANA,ANI.

Ці команди застосовуються для інвертування визначених бітів числа за допомогою числа-маски.

Крім того, за допомогою цих команд можна перевірити операнди на абсолютну рівність. У випадку рівності операндів результат буде вміщувати нулі в усіх розрядах.

Приклад 3

Інвертувати молодші біти числа, що знаходиться в акумуляторі.

Розв'язок: XRI 00001111B

Приклад 4

Перевірити рівність чисел, записаних у регістрах B і C. Результат залишити в акумуляторі.

Розв’язок:

MOV A, B ;Передавання числа з регістра B в акумулятор.

XRA C ;Виконання операції нерівнозначності над вмістом акумулятора і регістра C. У випадку рівності їх,в усіх бітах'акумулятора встановляться нулі.

Команда CMP

Ця команда служить для порівняння чисел, одне з яких .записане в акумуляторі, друге знаходиться в регістрі загального призначення або адресується другорядно регістровою парою H,L. Довжина команди 1 байт. При виконанні операції другий операнд віднімається від вмісту - акумулятора. Але на відміну від команди віднімання SUB результат, в акумуляторі не фіксується, тобто його вміст не змінюється. Результатом операції є значення регістра ознак (ругістра прапорців). Так, ознака Z (нуль) встановлюється в 1, якщо операнди рівні.

Команда CPI

Слід пам'ятати відмінність цієї команди від команди CMP. Тут другий операнд вказується беспосередньо в другому біті.

Приклад 5

Перевірити рівність чисел, записаних в регістрах B і C.

Розв’язок:

MOV A,B ;Передавання вмісту регістра B в акумулятор.

CMP C ;Порівняння чисел записаних в регістрах A і C.Вміст регістра C віднімається від вмісту : акумулятора, але вміст

акумулятора не змінюється.Ящо чісла що порівнюються рівні, ознака "нуль" Сприйме значення "1".

КОМАНДИ ЗСУВУ

1. Команди RLC, RRC

Ці команди виконують циклічний зсув вмісту акумулятора вліво (RLC) та вправо (RRC). При цьому біт, що висувається, розміщується також в розряді переносу регістра прапорців (рис. 1).

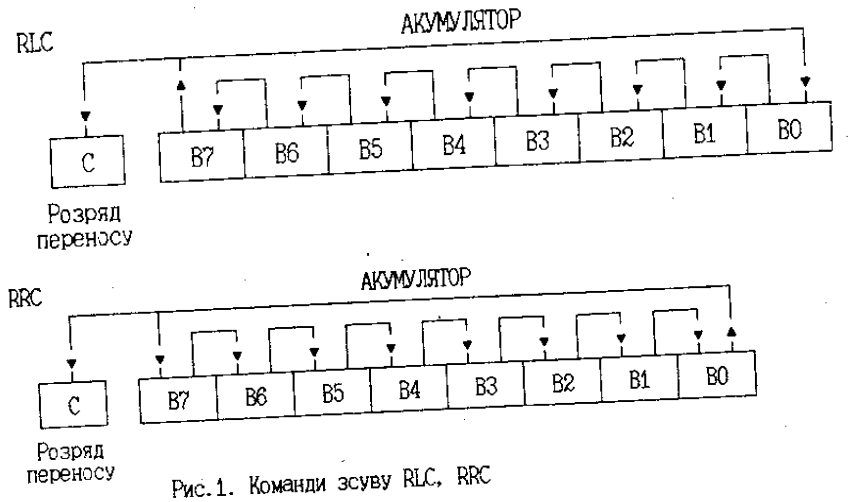


Рис.1. Команди зсуву RLC, RRC

Команди RAL і RAR

Ці команди виконують зсув вмісту акумулятора вліво CRAL) та вправо (RAR) через розряд ПЕРЕНОС, куди заноситься біт, що висувається. При цьому значення ознаки ПЕРЕНОС переходить в біт, що звільнився (рис. 2).

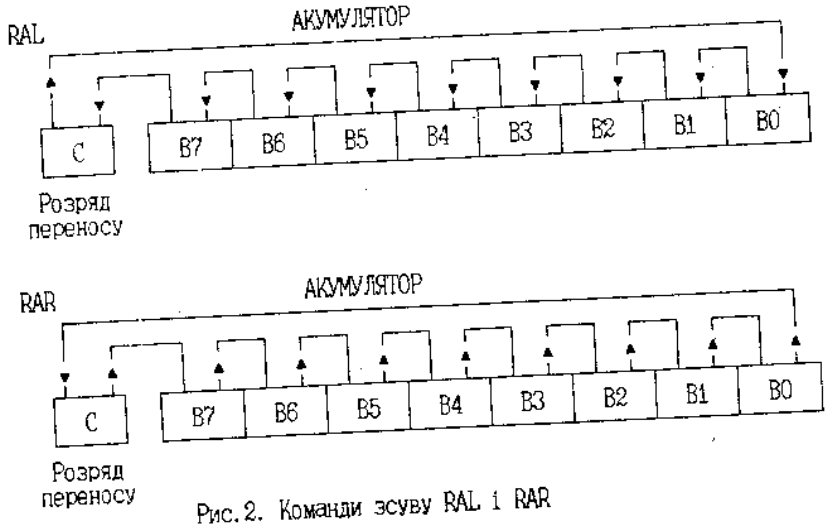


Рис. 2. Команди зсуву RAL і RAR

Кманда CMA

Це команда дає змогу інвертувати вміст акумулятора. Ознаки не діють, команда однобайтова.

Команда CMC

Ця команда йає змогу інвертувати значення розряда "ПЕРЕНОС" регістра прапорців (ознак).

Команда STC

Ця команда встановлює значення розряда "ПЕРЕНОС" регістра прапорців (ознак) в "1".

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ

Завдання для підготовки до роботи в лабораторії (самопідготовка)

2.1. Вивчити склад і призначення логічних команд. Засвоїти відмінні особливості і форму запису кожної логічної команди.

2.2.Скласти програму рішення нескладної задачі з застосуванням логічних команд на мові асемблера.

2.3.Провести процес ручного асемблювання складеної програми.

2.4.Записати отриману після асемблювання програму у вигляді таблиці адрес та даних, підготувати її таким чином для завантаження в ОЗП.
2.5.Підготувати заготовку звіта з відображенням результатів за пунктами 2.1 - 2.4. Заготувати таблицю результатів роботи в лабораторії за пунктами завдання 2.

Завдання і методика виконання роботи

2.6.Увімкнути живлення контролера і підготувати його до роботи (провести тестування контролера). Подальше вивчення кожної з команд мікропроцесора проводиться у такому порядку:
1)відповідно до завдання, вказаного для вивчення команди, скласти' програми на мові, асемблера;
2)провести ручне асемблювання програм;
3)за допомогою команд монітора:
-отримані машинні команди і дані завантажити в ОЗП;
-завантажити дані в регістри мікропроцесора;
-виконати програму;
-провести перевірку виконання програми.
Порядок роботи за кожним пунктом завдання і результати виконання кожної команди занести в підготовлену в процесі самостійної роботи заготовку звіта (таблицю результатів).

2.7. Дослідити застосування команд порозрядного логічного множення операндів ANA, ANI.

Примітка: у наступних завданнях адреси та виконання програм надані з урахуванням ОЗП та команд монітора для контролера програмованого універсального "Електроніка МС2721".

Завдання 1

Виконати операцію логічного множення над числами, наприклад, . ААН і 55Н. Перше число міститься в ОЗП в комірці з адресою 1А00Н, друге - в комірці з адресою 1А01Н. Виділити значення п'ятого біта результату і записати його в комірку 1А02Н.
Програму подано у вигляді таблиці 2.

Таблиця 2

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
	АДРЕСА	ДАНІ	
LXI H,1A00H			Запис адреси в регістрову пару H,L

MOV A,M			Передача першого операнда з комірки з адресою 1A00H в акумулятор
I NX H			Збільшення вмісту регістрової пари H, L (обчислення адреси наступного операнда)
ANA M			Логічне множення числа, записаного в акумуляторі, і числа з комірки з адресою 1A01H
ANI 001000Q0B			Виділення п’ятого біта результату
I NX H			Збільшення вмісту регістрової пари H,L на 1 (обчислення адреси результату)
MOV M,A			Передача вмісту акумулятора в комірку пам’яті з адресою 1A02H
HLT			Зупинка програми

Асемблювати і виконати програму самостійно.
При підготовці виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення чисел (наприклад, AAH і 55H) в ОЗП в комірки пам'яті з адресами IAOOH та 1A01H;
- запуск програш;
- контроль результату виконання.

Завдання 2
Виконати операцію логічного множення над числами (наприклад, 55FFH і FF55H). Виділити (роздільно) значення третього і сьомого бітів результатів множення, записати в пам’ять та проконтролювати результати.
Відповідно до завдання 2 самостійно скоректувати програму завдання 1 і за допомогою команд монітора виконати її аналогічно завданню 1.
Програму записати у вигляді таблиці. Результат занести у звіт.
Скласти, іасемблювати і виконати програму самостійно.
При підготовці виконання програми необхідно передбачити:

- занесення програми в. ОЗП ;
- занесення чисел (наприклад, 55FFH і FF55H) в ОЗП в -комірки пам’яті:
- запуск програми;
- контроль результату виконання.

2.8. Дослідити застосування команд порозрядного логічного додавання операндів ORA, 0RI.

Завдання 3

Встановити молодші біти числа (наприклад, 5АН), яке міститься в акумуляторі, в стан ”1”.

Програму подано у вигляді таблиці 3.

Таблиця 3

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
	АДРЕСА	ДАНІ	
OR I 00001111B			Установка 4-х молодших бітів числа в акумуляторі у стан "1”
HLT			Зупинка програми

Асемблювати і виконати програму самостійно.

При підготовці виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення числа (наприклад, 5АН) в акумулятор;
- запуск програми;
- контроль результату виконання.

Завдання 4

Об’єднати в одному байті старшого біта першого числа (наприклад, ААН) з молодшими бітами другого числа (наприклад, 55Н). Перше число міститься в ОЗП в комірці з адресою 1910Н; друге - в регістрі В. Результат занести в комірку ОЗП з адресою 1911Н.

Програму подано у вигляді таблиці 4.

Таблиця 4

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
	АДРЕСА	ДАНІ	
LDA 1910H			Передача першого числа з комірки з адресою 1910H в акумулятор
ANI 11110000B			Встановлення в стан "0" молодших бітів 1-го числа в акумуляторі
MOV C,A			Передача результату в регістр С
MOV A,B	•		Передача 2-го числа з регістра В в акумулятор
ANI 00001111B			Встановлення в стан "0" старших бітів 2-го числа в акумуляторі
ORA C			Логічне додавання числа в акумуляторі з вмістом регістра С
STA 1911H			Запис результату з акумулятора в комірку за адресою 1911H
HLT			Зупинка програм

Асемблювати і виконати програму самостійно.

При підготовці виконання програми необхідно передбачити;

- занесення програми в ОЗП;
- занесення першого числа (наприклад, AAH) в комірку ОЗП з адресою 1910H;
- занесення другого числа (наприклад, 55H) в регістр В;
- запуск програми;
- контроль результату виконання.

2.9. Дослідити застосування команд, порозрядно виконуючи операції нерівнозначності ЖА, ХШ.

Завдання 5

Інвертувати старший і три молодших біта числа (наприклад, 5АН), що знаходяться в акумуляторі.

Програму подано у вигляді таблиці 5.

Таблиця 5

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
	АДРЕСА	ДАНІ	
XRI 10000111B			Інвертування старшого і трьох молодших бітів чисел в акумуляторі
HLT			Зупинка програми

Асемблювати і виконати програму самостійно.

При підготовці виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення числа (наприклад, 5АН) в акумулятор;
- запуск програми;
- контроль результату виконання.

Завдання 6

Перевірити рівність чисел, які росташовуються в пам'яті (наприклад, за адресами 1920Н, 1921Н). Результат залишити в акумуляторі.

Програму подано у вигляді таблиці 6.

Таблиця 6

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
	АДРЕСА	ДАНІ	
LXI H, 1920H			Запис адреси першого операнда в регістрову пару H, L

M0V A,M			Передача першого операнда в акумулятор
Ш H,192FH			Запис адреси другого операнда в регістрову пару H, L
XRA M			Виконання операції нерівнозначності (визначення рівності чисел)
HLT			Зупинка програми

Асемблювати і виконати програму самостійно.
При підготовці виконання програми необхідно передбачити:

- занесення програми в ОЗП;
- занесення чисел в комірки пам'яті (наприклад, за адресами 1920H, 1921H);
- запуск програми;

- контроль результату виконання.

2.10. Дослідити застосування-команд порівняння даних CMP, CPI.

Завдання 7
Перевірити рівність двох однакових чисел (наприклад, 5AH), записаних у регістрах B і C.
Програму подано у вигляді таблиці 7.

Таблиця 7

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
	АДРЕСА	ДАНІ	
MOV A, B			Передача числа з регістра B в акумулятор
CMP C			Порівняння вмісту акумулятора з вмістом регістра C. Вміст акумулятора не змінюється. Результат зображається в бітах регістра ознак
HLT			Зупинка програми

Асемблювати і виконати програму самостійно.
При підготовці виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення числа (наприклад, 5AH) в регістри B і C;
- запуск програми;
- контроль результату виконання.

Звернути увагу на те, що результат зберігається в регістрі ознак, вміст акумулятора не змінюється.
Завдання 8
Перевіріти рівність чисел, записаних в регістрах B і C, коли друге число більше (менше) першого (наприклад, числа A6H, A5H).
Програму подано у вигляді таблиці 7.
Для виконання завдання 8 в регістри B і C потрібно занести вказанні числа і повторити виконання програми аналогічно завданню 7.
Проконтролювати новий вміст бітів регістра ознак. Результат занести у звіт.

2.11. Дослідити застосування команди отримання зворотного кода вмісту акумулятора CMA.
Завдання 9
Отримати додатковий код числа (наприклад, числа 75H), яке зберігається в регістрі C. Результат помістити в комірку ОЗП з адресою 1930H.
Програму подано у вигляді таблиці 8. .

Таблиця 8

АСЕМБЛЕР	ОБ'ЄКТНІ КОДИ		КОМЕНТАР
	АДРЕСА	ДАНІ	
MOV A,C			Передача числа в акумулятор
CMA			Отримання зворотнього коду числа в акумуляторі -
INR A			Отримання додаткового коду (збільшення числа в акумуляторі на 1)
LXI H,1930H			Занесення адреси комірки ОЗП в регістрову пару HI
MOV M, A			Передача результату з акумулятора в комірку ОЗП з адресою 1930H
HLT			Зупинка програми

Асемблювати і виконати програму самостійно.
При підготовці виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення числа (наприклад, 75H) в регістр C;
- запуск програми;

- контроль результату виконання.

2.12. Дослідити застосування команд зсуву вмісту акумулятора RLC, RRC, RAL, RAR.

Завдання 10

Дослідити застосування команд RLC, RRC, RAL, RAR шляхом зсуву вмісту акумулятора (нехай, наприклад, в акумуляторі міститься число 5АН).

Скласти, асемблювати і виконати програми самостійно.

При підготовці виконання програм необхідно передбачити:

занесення програм в ОЗП ;

- занесення числа (наприклад, 5АН) в акумулятор;
- запуск програм;
- контроль результату виконання (індикацію вмісту регістрів А, F).

2.13. Дослідити застосування команди інвертування значення розряду "перенос" регістра прапорців (ознак) СМС.

Завдання 11

Інвертування значення розряду "перенос" регістра прапорців (нехай, наприклад, у регістрі ознак міститься число 7Ш).

Скласти, асемблювати і виконати програму самостійно.

При підготовці виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення числа (наприклад, 7ВД в регістр прапорців;
- запуск програми;
- контроль результату виконання (індикацію вмісту регістра прапорців).

2.14. Дослідити застосування команди встановленя розряду "перенос" регістра прапорців (ознак) в стан "1" БТС.

Завдання 12

Встановити розряд "перенос" регістра ознак в стан "1" (нехай, наприклад, у регістрі прапорців міститься число 7ЕН).

Скласти, асемблювати і виконати програму самостійно.

При підготовці виконання програми необхідно передбачити:

- занесення програми в ОЗП ;
- занесення числа (наприклад, 7ЕН) в регістр прапорців;
- запуск програми;
- контроль результату виконання (індикацію вмісту регістра прапорців).