МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання самостійних робіт
з курсу: «**Архітектура обчислювальних систем»**

ЛАБОРАТОРНИЙ ПРАКТИКУМ З АРХІТЕКТУРИ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

для студентів спеціальностей:
122 «Комп'ютерні науки»
113 «Прикладна математика»

затверджено
редакційно-видавничою
радою університету,
протокол №1 від06.02.2023р.

Харків
НТУ «ХПІ»
2023

Методичні вказівки для виконання лабораторних робіт з курсу «Архітектура обчислювальних систем» для студентів спеціальностей: 122 «Комп’ютерні науки», 113 «Прикладна математика» / уклад.: Водка О.О., Литвинов А.Л, Марусенко О.М., Трубаєв О.І., Під’ячий Г. - Харків.: НТУ «ХПІ», 2023. 45 с.

Укладачі: О. О. Водка
А. Л. Литвинов,
О. М. Марусенко, ,
О. І. Трубаєв
Г. Ю. Під’ячий

Рецензент проф. А.О. Зуєв
Кафедра автоматики і управління в технічних системах

ВСТУП

Методичні вказівки з архітектури обчислювальних систем призначені для студентів, які вивчають основи комп'ютерних наук та інформаційних технологій. Цей курс розроблено з метою розкриття фундаментальних аспектів будови та функціонування обчислювальних систем, що лежать в основі сучасних комп'ютерів.

Кожна робота в методичних вказівках включає в себе практичні завдання та експерименти, спрямовані на розуміння різних аспектів обчислювальних систем. Ви будете досліджувати такі важливі теми, як системи числення, центральні та зовнішні пристрої персональних комп'ютерів, моделювання електронних схем у програмі Multisim, функціонування блоків живлення стаціонарних комп'ютерів, логічні схеми та схеми пам'яті на діодах і транзисторах, тригери комп'ютерних систем, лічильники, арифметико-логічні пристрої та програмування на мові Assembler.

Цей курс допоможе поглибити знання про те, як працюють обчислювальні системи на найнижчому рівні, що є важливим для розуміння принципів їх роботи та для розвитку навичок у галузі обчислювальної техніки. Кожна робота включає в себе теоретичний матеріал та практичні завдання, що сприятимуть кращому розумінню та вмінням у сфері архітектури обчислювальних систем.

Бажаємо вам успішної роботи та плідного навчання у галузі архітектури обчислювальних систем!

РОБОТА №1. СИСТЕМИ ЧИСЛЕННЯ

Мета роботи: вивчення позиційних систем числення.

1.1. Короткі теоретичні відомості

Система числення – символічний метод запису чисел, подання чисел за допомогою письмових знаків [1, 2].

Число – одне з основних понять математики, що використовується для кількісної характеристики, порівняння, нумерації об'єктів та їх частин.

Цифри – це символи, які використовуються для запису чисел.

Оскільки чисел набагато більше, ніж цифр, то для запису числа зазвичай використовується набір (комбінація) цифр.

Існує багато способів запису чисел за допомогою цифр. Кожен такий спосіб називається системою числення. Величина числа може залежати від порядку цифр у записі, а може й не залежати. Ця властивість визначається системою числення і є основою найпростішої класифікації таких систем. Зазначений факт дозволяє всі системи числення розділити на три класи (групи): позиційні, непозиційні та змішані.

Прикладом непозиційної системи числення є римська система, де величина, яку позначає цифра, не залежить від позиції її у числі. Найвідомішим прикладом змішаної системи числення є грошова система та представлення часу у вигляді кількості різних одиниць часу – діб, годин, хвилин і секунд. При цьому величина d днів h годин m хвилин s секунд відповідає значенню часу T у секундах, яке підраховується за формулою

$$T=D\cdot 24\cdot 60\cdot 60+h\cdot 60\cdot 60+m\cdot 60+s \text{ секунд.} \qquad (1.1)$$

Позиційні системи числення – це системи числення, у яких значення цифри безпосередньо залежить від її позиції в числі. Наприклад, число 01 означає одиницю, 10 – десять. Позиційні системи числення дозволяють легко проводити арифметичні розрахунки.

Подання чисел з допомогою арабських цифр використовується в багатьох системах числення. Найпоширенішою серед них є позиційна система числення, яка називається десятковою.

Для складання машинних кодів зручно використовувати не десяткову, а двійкову систему числення, що містить лише дві цифри 0 і 1. Програмісти для обчислень також користуються вісімковою і шістнадцятковою системами числення.

У кожній системі числення є своя основа – число за ступенями якого записуються всі числа у цій системі. У деяких джерелах з основою асоціюють кількість цифр, що використовуються в системі числення, але таке визначення мабуть не є коректним. У десятковій системі основа дорівнює десяти, у двійковій системі – двом, а у вісімковій та шістнадцятковій – відповідно, восьми та шістнадцяти. У q-ковій системі числення кількість цифр дорівнює q і використовуються цифри від 0 до q – 1.

Для роботи з різними системами числення необхідно знати представлення десяткових чисел від нуля до 15 у системах числення з основами q = 2, 8, 16 (див. табл. 1.1). Крім цього, корисно знати десяткові значення чисел $2k$ для k = 0, 1,…, 10 (див. табл. 1.2).

Переведення чисел з однієї системи числення до іншої

Для перекладу цілого числа N з q-ковою основою в десяткове число записують у вигляді многочлена, а потім обчислюють його за правилами десяткової арифметики:

$$N = a_n \cdot q^n + a_{n-1} \cdot q^{n-1} + \ldots + a_2 \cdot q^1 + a_1 \cdot q_0. \qquad (1.2)$$

де a_n – це цифри у запису числа,

q – основа системи числення,

n = 0, 1, 2 ...

Таблиця 1.1–Подання десяткових чисел у різних системах числення

q = 10	q = 2	q = 8	q = 16
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

Таблиця 1.2 – Значення чисел 2^k

k	0	1	2	3	4	5	6	7	8	9	10
2^k	1	2	4	8	16	32	64	128	256	512	1024

Приклад:

$$(11001)_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 =$$
$$= 1 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = (25)_{10}$$

$$(221)_3 = 2 \cdot 3^2 + 2 \cdot 3^1 \cdot 1 \cdot 3^0 = 2 \cdot 9 + 2 \cdot 3 + 1 \cdot 1 = (25)_{10}$$
$$(31)_8 = 3 \cdot 8^1 + 1 \cdot 8^0 = 3 \cdot 8 + 1 \cdot 1 = (25)_{10}$$

$$(524D)_{16} = 5 \cdot 16^3 + 2 \cdot 16^2 + 4 \cdot 16^1 + 13 \cdot 16^0 =$$
$$= 20480 + 512 + 64 + 13 = (21069)_{10}$$

Примітка: під час роботи з різними системами числення число записують у дужках, а за дужками – основу системи.

Для зворотного перетворення цілих чисел (з десяткової системи числення в систему з основою q) число N ділять на q і записують залишки від розподілу до тих пір, поки частка від попереднього розподілу не буде рівною нулю.

Приклад: перетворимо число 25 у двійкову систему:

Таблиця 1.3 – Перетворення числа 25

Початкове число	частка	залишок
25/2	12	1
12/2	6	0
6/2	3	0
3/2	1	1
1/2	0	1

Коли остання частка стала рівною нулю, записують всі залишки поспіль від останнього до першого. Таким чином, отримали число в двійковій системі числення. Результат: $(25)_{10}=(11001)_2$.

Для перекладу змішаних чисел у двійкову систему числення потрібно окремо переводити їхню цілу та дробову частини. У записі результату ціла частина перекладу відокремлюється від дробової коми відповідно до формули:

$$N = \pm a_n a_{n-1}...a_2 a_1 a_0 , a_{-1} a_{-2}...a_{-n} . \quad (1.3)$$

Основні системи числення:

– Двійкова система числення. У комп'ютерній техніці переважно використовується двійкова система числення. Таку систему дуже легко реалізувати в цифровій мікроелектроніці, так як для неї потрібно всього два стійкі стани (0 і 1).
– Вісімкова система числення. Позиційна цілочислова система числення з основою 8. Для представлення чисел у ній використовуються цифри від 0 до 7. Вісімкова система числення часто використовується в технічних галузях, пов'язаних з цифровими пристроями. Характеризується легким переведенням вісімкових чисел у двійкові та зворотно, шляхом заміни вісімкових чисел на тріади двійкових. Раніше ця система широко використовувалася у програмуванні та комп'ютерній документації, проте в даний час майже повністю витіснена шістнадцятковою системою. Для переведення двійкового нецілого числа у вісімкове вихідне число розбивають на тріади вліво та вправо від коми; відсутні крайні цифри доповнюють нулями. Потім кожну тріаду записують восьмеричною цифрою (див. табл. 1.1). Приклад: ілюстрація переведення двійкового числа до восьмеричного числа:

$$N=(110011,100010)_2=(110\ \ 011,100\ \ 010)_2=(63,42)_8$$

– Шістнадцяткова система числення. Позиційна система числення з основою 16. Зазвичай як шістнадцяткові цифри використовуються десяткові цифри від 0 до 9 і латинські літери від A до F для позначення чисел від $(10)_{10}$ до $(15)_{10}$, тобто (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)$_{16}$.

Для перекладу двійкового числа в шістнадцяткове вихідне число розбивають на тетради вліво і вправо від коми; відсутні крайні цифри доповнюють нулями. Потім кожну тетраду записують шістнадцятковою цифрою (див. табл. 1.2).

Приклад: ілюстрація переведення двійкового числа до шістнадцяткового числа

$$N=(0111\ 1010\ 1011,1110\ 1111)_2=(7AB,EF)_{16}$$

1.2. Порядок виконання роботи

Виконати завдання до лабораторної роботи. Варіанти завдань наведені у табл. 1.4.

Завдання 1. Перевести цілі числа із десяткової системи числення:

– у двійкову;

– у вісімкову;

– у шістнадцяткову.

Завдання 2. Перевести цілі числа із двійкової системи числення:

– у вісімкову;

– у шістнадцяткову;

– у десяткову.

Завдання 3. Перевести цілі числа із шістнадцяткової системи числення:

– у двійкову;

– у вісімкову;

– у десяткову.

Завдання 4. Скласти:

– двійкові числа;

– вісімкові числа;

– шістнадцяткові числа.

Завдання 5. Знайти різницю:

– двійкових чисел;

– вісімкових чисел;

– шістнадцяткових чисел.

Завдання 6. Обчислити значення висловлювання і записати у десятковій системі числення.

Таблиця 1.4 – Варіанти завдань до лабораторної роботи

№ завдання	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5
Завдання 1	a) 2515 b) 3074 c) 9042	a) 1052 b) 1377 c) 7634	a) 2042 b) 5547 c) 2372	a) 5911 b) 6321 c) 7629	a) 3988 b) 5147 c) 1123
	Варіант 6	Варіант 7	Варіант 8	Варіант 9	Варіант 10
	a) 1415 b) 4074 c) 9942	a) 2152 b) 2477 c) 8734	a) 3042 b) 6647 c) 3472	a) 6911 b) 7421 c) 8729	a) 4088 b) 6247 c) 2423
	Варіант 11	Варіант 12	Варіант 13	Варіант 14	Варіант 15
	a) 2504 b) 3073 c) 9031	a) 1041 b) 1376 c) 7623	a) 2031 b) 5559 c) 2361	a) 5901 b) 6319 c) 7618	a) 3977 b) 5136 c) 1112
	Варіант 16	Варіант 17	Варіант 18	Варіант 19	Варіант 20
	a) 2625 b) 3074 c) 9152	a) 1162 b) 1377 c) 7744	a) 2152 b) 5547 c) 2482	a) 5911 b) 6321 c) 7739	a) 3988 b) 5157 c) 1313
	Варіант 21	Варіант 22	Варіант 23	Варіант 24	Варіант 25
	a) 2616 b) 3175	a) 1153 b) 1477	a) 2143 b) 5647	a) 5812 b) 6421	a) 3889 b) 5257

	c) 9143	c) 7735	c) 2473	c) 7728	c) 1224
Завдання 2	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5
	a) 110101 b) 111010 c) 101111	a) 011001 b) 101010 c) 010101	a) 100110 b) 110011 c) 101111	a) 011011 b) 101011 c) 010111	a) 100010 b) 111000 c) 011111
	Варіант 6	Варіант 7	Варіант 8	Варіант 9	Варіант 10
	a) 100101 b) 101010 c) 111111	a) 001001 b) 111010 c) 011101	a) 110110 b) 100011 c) 111111	a) 001011 b) 111011 c) 110111	a) 110010 b) 101000 c) 101111
	Варіант 11	Варіант 12	Варіант 13	Варіант 14	Варіант 15
	a) 101101 b) 100010 c) 110111	a) 011001 b) 110010 c) 010101	a) 110010 b) 100111 c) 111011	a) 001111 b) 110111 c) 110011	a) 110110 b) 101100 c) 101011
	Варіант 16	Варіант 17	Варіант 18	Варіант 19	Варіант 20
	a) 101011 b) 100001 c) 111101	a) 011111 b) 110100 c) 110111	a) 110100 b) 110101 c) 111101	a) 101101 b) 110111 c) 110101	a) 110110 b) 101110 c) 101011
	Варіант 21	Варіант 22	Варіант 23	Варіант 24	Варіант 25
	a) 101111 b) 100000 c) 110101	a) 011011 b) 110000 c) 010111	a) 110000 b) 100101 c) 111001	a) 001101 b) 110101 c) 110001	a) 110100 b) 101110 c) 101001
Завдання 3	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5
	a) 2F52 b) 4521 c) 2101	a) 2A1B b) 3350 c) 4239	a) 6EE2 b) 3672 c) 3461	a) 8B1B b) 4457 c) 7537	a) 2C2D b) 7724 c) 9673
	Варіант 6	Варіант 7	Варіант 8	Варіант 9	Варіант 10
	a) 1F52 b) 5521 c) 1101	a)1A1B b) 2350 c) 3239	a) 5EE2 b) 2672 c) 2461	a) 7B1B b) 3457 c) 6537	a) 1C2D b) 6724 c) 8673
	Варіант 11	Варіант 12	Варіант 13	Варіант 14	Варіант 15
	a) 3F52 b) 6521 c) 3101	a) 3A1B b) 4350 c) 5239	a) 4EE2 b) 1672 c) 1461	a) 6B1B b) 2457 c) 5537	a) 3C2D b) 7724 c) 7673
	Варіант 16	Варіант 17	Варіант 18	Варіант 19	Варіант 20
	a) 4F52 b) 7521 c) 4101	a) 4A1B b) 5350 c) 6239	a) 3EE2 b) 4672 c) 4461	a) 5B1B b) 1457 c) 4537	a) 4C2D b) 6724 c) 6673
	Варіант 21	Варіант 22	Варіант 23	Варіант 24	Варіант 25
	a) 1A52 b) 5B21 c) 1C01	a) 1B1B b) 2450 c) 3339	a) 5EB2 b) 26A2 c) 2A61	a) 7A1B b) 3958 c) 6C37	a) 1A2D b) 6B24 c) 8A73
Завдання 4	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5
	a) 1011+0111 b) 573+325 c) F1 + E7	a) 0110+1100 b) 274+235 c) 93 + 2C	a) 1010+0101 b) 271+123 c) 58 + 79	a) 1101+1101 b) 632 + 714 c) 51 + 9D	a) 1010+1010 b) 521+623 c) 36 + AB
	Варіант 6	Варіант 7	Варіант 8	Варіант 9	Варіант 10

	a) 0011+0111 b) 673+325 c) E1 + E7	a) 1110+1100 b) 174+235 c) 83 + 2C	a) 0010+0101 b) 171+123 c) 48 + 79	a) 0101+1101 b) 532+714 c) 41 + 9D	a) 0010+1010 b) 421+623 c) 26 + AB
	Варіант 11	Варіант 12	Варіант 13	Варіант 14	Варіант 15
	a) 1111+0111 b) 563+325 c) F2 + E7	a) 0010+1100 b) 264+235 c) 94 + 2C	a) 1110+0101 b) 261+123 c) 59 + 79	a) 1001+1101 b) 622+714 c) 52 + 9D	a) 1110+1010 b) 531+623 c) 37 + AB
	Варіант 16	Варіант 17	Варіант 18	Варіант 19	Варіант 20
	a) 1001+0111 b) 574+325 c) F2 + A7	a) 0100+1100 b) 275+235 c) 94 + 2C	a) 1000+0101 b) 272+123 c) 59 + 79	a) 1100+1101 b) 633+714 c) 52 + 9D	a) 1000+1010 b) 522+623 c) 35 + AB
	Варіант 21	Варіант 22	Варіант 23	Варіант 24	Варіант 25
	a) 1010+0111 b) 371+325 c) B1 + C7	a) 0111+1100 b) 476+225 c) 72 + 2A	a) 1010+0111 b) 176+213 c) 78 + 78	a) 1101+1111 b) 662+774 c) 21 + 7D	a) 1010+1011 b) 321+733 c) 26 + CB
Завдання 5	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5
	a) 1010–0111 b) 371–325 c) B1 – C7	a) 0111–1100 b) 476– 225 c) 72 – 2A	a) 1010–0111 b) 176– 213 c) 78 – 58	a) 1101–1111 b) 662 – 774 c) 21 – 7D	a) 1010–1011 b) 321– 733 c) 26 – CB
	Варіант 6	Варіант 7	Варіант 8	Варіант 9	Варіант 10
	a) 1011–0111 b) 573– 325 c) F1 – E7	a) 1110–1100 b) 274– 235 c) 93 – 2C	a) 1010–0101 b) 271– 123 c) A8 – 79	a) 1101–1001 b) 732 – 714 c) B1 – 9D	a) 1010–1000 b) 721 –623 c) C6 – AB
	Варіант 11	Варіант 12	Варіант 13	Варіант 14	Варіант 15
	a) 1111–0111 b) 563– 325 c) F2 – E7	a) 0010–1100 b) 264– 235 c) 94 – 2C	a) 1110–0101 b) 261– 123 c) 59 – 79	a) 1001–1101 b) 622 – 714 c) 52 – 9D	a) 1110–1010 b) 531– 623 c) 37 – AB
	Варіант 16	Варіант 17	Варіант 18	Варіант 19	Варіант 20
	a) 1111–0111 b) 563– 325 c) F2 – E7	a) 0010–1100 b) 264– 235 c) 94 – 2C	a) 1110–0101 b) 261– 123 c) 59 – 79	a) 1001–1101 b) 622 – 714 c) 52 – 9D	a) 1110–1010 b) 531– 623 c) 37 – AB
	Варіант 21	Варіант 22	Варіант 23	Варіант 24	Варіант 25
	a) 1001–0111 b) 574– 325 c) F2 – A7	a) 0100–1100 b) 275– 235 c) 94 – 2C	a) 1000–0101 b) 272– 123 c) 59 – 79	a) 1100–1101 b) 633 – 714 c) 52 – 9D	a) 1000–1010 b) 522– 623 c) 35 – AB
Завдання 6	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5
	23_8+$A2_{16}$ *1001_2	$B1_{16}$–1011_2 * 117_8	51_8 * 21_{16} – 455_{10}	(59_{16} + +1110_2)*456_8	25_8 *567_{16} – 10101_2
	Варіант 6	Варіант 7	Варіант 8	Варіант 9	Варіант 10
	23_8 – $A3_{16}$ * 1011_2	$B2_{16}$+1001_2 * 117_8	57_8 * 21_{16} + 455_{10}	(59_{16} – –1110_2)*46_8	45_8 *547_{16} + 10101_2

	Варіант 11	Варіант 12	Варіант 13	Варіант 14	Варіант 15
	$34_8 + B3_{16}$ $* 1001_2$	$C2_{16} – 1011_2$ $* 126_8$	$62_8 * 32_{16}$ $– 465_{10}$	$(60_{16} +$ $+1110_2)*476_8$	$14_8 *346_{16}$ $– 10001_2$
	Варіант 16	Варіант 17	Варіант 18	Варіант 19	Варіант 20
	$45_8 + C4_{16}$ $* 1011_2$	$E4_{16} – 1111_2$ $* 117_8$	$53_8 * 32_{16} –$ $– 355_{10}$	$(48_{16} +$ $+1010_2)*456_8$	$34_8 *324_{16}$ $– 10101_2$
	Варіант 21	Варіант 22	Варіант 23	Варіант 24	Варіант 25
	$56_8 + F2_{16}$ $* 1101_2$	$D1_{16} – 1000_2$ $* 126_8$	$40_8 * 21_{16}$ $– 455_{10}$	$(48_{16} +$ $+1100_2)*156_8$	$25_8 *132_{16}$ $– 11101_2$

Зміст звіту:

Короткий опис теоретичних відомостей.

Опис виконання практичної частини роботи. У звіті обов’язково детально відобразити процес перетворення чисел з однієї системи числення до іншої.

Письмові відповіді на контрольні питання.

Висновки.

Контрольні питання:

1. Що називається системою числення?

2. Які системи числення називаються непозиційними? Чому? Наведіть приклад такої системи числення та запису чисел у ній.

3. Які системи числення застосовуються у обчислювальній техніці: позиційні чи непозиційні? Чому?

4. Як зображається число у позиційній системі числення?

5. Що називається основою системи числення?

6. Як можна уявити ціле позитивне число у позиційній системі числення?

7. Які системи числення застосовуються на комп'ютері для представлення інформації?

8. За якими правилами виконується складання двох позитивних цілих чисел?

9. Які правила виконання арифметичних операцій у двійковій системі числення?

10. Для чого використовується переведення чисел з однієї системи числення до іншої?

11. Сформулюйте правила переведення чисел із системи числення з основою p в десяткову систему числення і назад: із десяткової системи числення в систему числення з основою s. Наведіть приклади.

12. Як виконати переведення чисел із двійкової системи числення у вісімкову систему і назад? З двійкової системи числення в шістнадцяткову систему і навпаки? Наведіть конкретні приклади.

13. За якими правилами виконується переведення чисел з вісімкової в шістнадцяткову систему числення та навпаки? Наведіть приклади.

РОБОТА №2. ЦЕНТРАЛЬНІ І ЗОВНІШНІ ПРИСТРОЇ ПК

Мета роботи: вивчення функціональної взаємодії центральних та зовнішніх пристроїв комп'ютера та їх характеристик.

Прилади та приладдя: комплектуючі елементи системного блоку: корпус системного блоку, блок живлення, материнська плата, процесор, оперативна пам'ять, відеокарта, жорсткий диск. Блок CD/DVD-ROM, монітор, клавіатура, миша. Кулер, термопаста, викрутка плоска, викрутка хрестова, комп'ютерні болти.

2.1. Теоретичні відомості

Персональний комп'ютер (ПК) настільного виконання в мінімальній конфігурації складається з таких складових частин: системний блок, монітор, клавіатура, маніпулятор-миша [3-6].

2.1.1. Корпус системного блоку є прямокутним паралелепіпедом, виготовленим з листового металу (сталь, алюмінієвий сплав), всередині якого розташовані всі компоненти, необхідні для функціонування ПК. Неважливо, яка компанія виготовила корпус комп'ютера, будь-який корпус комп'ютера відповідає єдиному стандарту, який називається формфактором.

Формфактор (form factor) – це стандарт технічного виробу, що описує сукупність його технічних параметрів, наприклад форму, розмір, положення та типи роз'ємів, вимог до вентиляції, напруги та інших параметрів.

Розрізняють два типи формфактора корпусу ПК: AT та ATX.

Формфактор AT (англ. «Advanced Technology» – передова технологія, клас PC на процесорах 286: технологія використовувалася для комп'ютерів, включаючи Pentium-1) – це перший формфактор, що широко використовувався в персональних комп'ютерах у 80-х роках минулого століття і в даний час більше не випускається. Формфактор AT був створений IBM у 1984 році і прийшов на зміну раніше існуючим форм-факторам PC та XT.

Формфактор ATX (від англ. Advanced Technology Extended» – передова технологія розширена) – формфактор персональних настільних комп'ютерів.

У 1995 році корпорація INTEL запропонувала нову специфікацію корпусу – систему блоку ATX (AT eXtention). Його поява обумовлена збільшеною потужністю персональних комп'ютерів та наявністю великої кількості зовнішніх пристроїв, що підключаються до ПК, а також вбудованою мікросхемою в материнську плату, що підвищило вимоги до охолодження цих елементів. Форм-фактор ATX є домінуючим форматом для масово-випущених комп'ютерів, починаючи з 2001 року.

Залежно від габаритних розмірів корпуси системних блоків поділяються на типи, перелік яких наведено у табл. 2.1.

Таблиця 2.1 – Розміри корпусів системних блоків ПК

Тип корпуса	Висота (см)	Ширина (см)	Довжина (см)
Desktop	20	45	45
Slimline	7	35	45
Mini Tower	45	20	45
Midi Tower	50	20	45
Big Tower	65	20	48
Super Big Tower	75	22	48
File Server	75	35	55

Наскільки це важливо – знати формфактор ПК? Від правильності вибору формфактора комп'ютера залежить безліч важливих параметрів: наприклад, рівень шуму, температура корпусу, вентиляція, які впливають на роботу комплектуючих внутрішніх частин системного блоку: насамперед жорсткого диска, центрального процесора, інших складових частин системного блоку.

Формфактор ATX визначає такі основні характеристики ПК:

– геометричні розміри материнських плат;

– загальні вимоги щодо положення роз'ємів та отворів на корпусі;

– положення блоку живлення в корпусі;

– геометричні розміри блоку живлення;

– електричні характеристики блоку живлення та кольору проводів;

– форму та положення ряду роз'ємів (переважно живлення).

Приклад: колір дротів стандарту ATX (див. табл. 2.2).

Таблиця 2.2 – Колір проводів стандарту ATX

Ланцюг	Колір дротів	Поянення
+5V	Червоний	Основна напруга
+12V	Жовтий	Живлення двигунів пристроїв та інтерфейсних ланцюгів
-5V	Білий	Не використовується. Присутні для дотримання стандарту ISA Bus
-12V	Синій	Живлення інтерфейсних ланцюгів
+3,3V	Помаранчевий	Живлення процесора
+3,3V Sence	Коричневий	Сигнал зворотного зв'язку стабіліза-тора +3,3V
+5VSB	Малиновий	Чергове малопотужне джерело +5V
PS-ON	Зелений	Сигнал увімкнення джерел живлен-ня
PW-OK	Сірий	Сигнал «живлення в нормі»
GND	Чорний	Загальний провід щодо напруги жи-влення

2.1.2. У корпусі системного блоку розміщуються такі компоненти:

- материнська (системна) плата;
- блок живлення;
- жорсткий диск;
- дисковод гнучких дисків;
- привід CD/DVD дисків;
- плати розширення (встановлюються в слотах материнської плати, а кріпляться до задньої стінки корпусу системного блоку).

Материнська плата є головною складовою персонального комп'ютера. Від її якості та швидкодії безпосередньо залежить стабільність і швидкість роботи всього комп'ютера.

Материнська плата визначає фундаментальні параметри комп'ютера (тип процесора, тип пам'яті), що визначають рівень продуктивності, а також практичні параметри (формфактор, кількість слотів розширення, наявність інтегрованих пристроїв), споживчі властивості та можливу сферу застосування.

2.1.3. На материнській платі розміщено таке:

- сокет для встановлення процесора;
- основний набір мікросхем материнської плати (чіпсет);
- мікросхема BIOS (Basic Input Output System);
– модулі ОЗП;
– слоти для встановлення плат розширення;
– конектори для підключення IDE- або SCSI-пристроїв та FDD;
– допоміжні конектори для підключення індикаторних елементів та кнопок керування, розташованих на передній панелі корпусу;
- зовнішні порти для клавіатури, миші, принтера, шини USB та ін.

Розміри материнської плати визначаються також форм-фактором. Основні їх формфактори перелічені у таблиці 2.3.

Таблиця 2.3 – Формфактори материнських плат

Форм фактор	Фізичні розміри (дюйми/мм)	Специфікація, рік	Примітка
XT	(8,5 x 11)″ (216x279)	IBM, 1983	Архітектура IBM PC XT
AT	(12 × 11 – 13)″ (305x279-330)	IBM, 1984	Архітектура IBM PC AT (Desktop/Tower)
ATX	(12 × 9,6)″ (305x244)	Intel, 1995	Для системних блоків типів MiniTower, FullTower
Mini ATX	(11,2 × 8,2)″ (284x208)	Intel, 1995	Для системних блоків типу Tower і компактних Desktop
Micro ATX	(9,6 × 9,6)″ (244x244)	Intel, 1997	Має менше слотів, ніж ATX

2.1.4. Безпосередньо на материнську плату встановлюються:

- процесор;
- радіатор процесора з кулером;
– модулі оперативної пам'яті;
– плати розширення (кріпляться гвинтами до задньої стінки системного блоку).

Такі модулі, як блок живлення, жорсткий диск, дисковод гнучких дисків, привод CD/DVD дисків, розміщуються у спеціальних відсіках корпусу системного блоку та кріпляться до нього гвинтами. Останнім часом набув популярності привід Blu-ray (BD-ROM). Це вдосконалена технологія оптичних носіїв, в основі якої лежить використання лазера з довжиною хвилі 405 нм. Такі привіди розміщуються окремо від корпусу системного блоку.

2.2. Порядок виконання роботи

Завдання 1. Аналіз архітектури робочого комп'ютера студента та збирання комп'ютера Вашої мрії

1. За допомогою безкоштовних системних інформаційних утиліт [7] провести аналіз архітектури робочого комп'ютера студента.

2. За допомогою програмного засобу запропонованого на сайті [8] зібрати комп'ютер мрії студента.

Додаткові завдання (виконуються при наявності необхідного обладнання за рекомендацією викладача)

Завдання 2. Ідентифікація компонентів системного блоку та материнської плати

Отримайте у викладача набір компонентів, призначених для збирання системного блоку ПК. Визначте форм-фактор корпусу ПК. Акуратно розкладіть компоненти на столі, визначте їх призначення та заповніть таблицю 2.4.

Таблиця 2.4 – Основні компоненти системного блоку ПК

№	Назва компоненти	Модель, тип, функціональне призначення
1	Формфактор корпусу	
2	Материнська плата	
3	Блок живлення	
4	Жорсткий диск	
5	Дисковод гнучких дисків	
6	Привід CD/DVD дисків	
7	Кулер	
8	Інше…	

Визначте розташування на материнській платі основних встановлених компонентів, а також компонентів, що на неї додатково встановлюються, та заповніть таблицю 2.5.

Таблиця 2.5 – Основні компоненти материнської плати ПК

№	Назва компонента	Встановлені компоненти (або потреба встановити)	Модель, тип, функціональне призначення
1	Soket для установки процесора		
2	Процесор		
3	Радіатор процесора з кулером		
4	Чіпсет		
5	Модулі оперативної пам’яті		
6	Мікросхема BIOS		
7	Слоти для плат розширення		
8	Плати розширення		
9	Зовнішні порти для клавіатури		
10	Зовнішні порти для миші		
11	Зовнішні порти для принтера		
12	Шини USB внутрішні		
13	Зовнішні порти для USB		
14	Інше…		

Завдання 3. Встановлення процесора та радіатора з кулером

Складання комп'ютера починається з установки процесора в сокет материнської плати. У сокету процесора на системній платі підніміть на 90° важіль фіксації контактів. Цей ричаг може бути пластмасовим або металевим.

Процесор піднесіть до сокету та визначте положення ключів-контактів, пропущених по кутах корпусу процесора та сокету. Процесор повинен встановлюватися в сокет без зусиль, інакше можна погнути контакти процесора.

Закрийте важіль сокету. Зусилля, що прикладається до важеля, не повинно бути надто великим.

Для охолодження процесора використовується радіатор із закріпленим на ньому кулером (вентилятором). Для підвищення ефективності передачі тепла від процесора радіатору застосовується термопаста (наприклад типу КПТ-8), яка наноситься на кришку процесора перед встановленням на нього радіатора. На центр кришки процесора з тюбика наноситься тонкий шар термопасти.

На процесор дуже акуратно накладіть радіатор і притисніть його до процесора. Закріпіть радіатор.

Завдання 4. Встановлення модулів пам'яті

Перед встановленням модулів пам'яті обов'язково слід знайти у документації на системну плату таблицю установки модулів по слотам.

Існують певні обмеження щодо комбінацій модулів у слотах. Зазвичай одиночний модуль повинен бути встановлений у перший слот першого банку пам'яті. Якщо чіпсет, на основі якого побудована системна плата, має двоканальну конфігурацію пам'яті, то для активації двоканальної пам'яті в різні банки пам'яті можна встановлювати лише однакові за структурою модулі пам'яті.

При установці модуль пам'яті вводиться в напрямні слота, при цьому слід простежити, щоб прорізи (ключі) у друкованій платі модуля пам'яті та пластмасові виступи на слоті збігалися. Клямки на слотах повинні бути відведені убік. Для остаточного встановлення модуля пам'яті треба сильно натиснути двома пальцями по краях модуля. При правильній установці клямки повинні зафіксувати модуль у слоті, у цьому випадку зазвичай пролунає негучне клацання. Якщо фіксації не сталося, то модуль при установці був неправильно орієнтований або робиться спроба встановити непідходящий для цієї системної плати модуль пам'яті.

Для двоканальних варіантів модулі пам'яті встановлюються у різні слоти (банки). Прийнято використовувати для слотів різнобарвну пластмасу, тому рекомендується встановлювати модулі у слоти однакового кольору.

Завдання 5. Монтаж материнської плати

Системна плата кріпиться в корпусі комп'ютера на шестигранних металевих різьбових втулках.

Шестигранні втулки вкручуються в тих точках кріплення на корпусі системного блоку, де є отвори з різьбленням. Металеві втулки використовуються не тільки для кріплення системної плати, але й для створення точок заземлення. Як правило, такі дві точки заземлення є на краю системної плати, де знаходяться слоти розширення. Для ефективного заземлення на системній платі навколо отворів кріплення (не обов'язково всіх) виконується струмопровідна доріжка. У тих точках кріплення, де на системній платі немає заземлювальної струмопровідної доріжки, при використанні металевих втулок слід додатково використовувати пластмасові шайби.

Визначившись із точками кріплення, встановіть системну плату в корпус. Якщо блок живлення розташований над поверхнею системної плати (як у корпусах Mini Tower), то переконайтеся, що радіатор процесора не впирається в блок живлення (залишається зазор близько 10 мм). Кріпити системну плату гвинтами слід лише після перевірки збігу всіх кріпильних стійок з отворами на системній платі, можливості вільної установки блока живлення та відсутності заглушок у корпусі навпроти слотів розширення, в які передбачається встановити плати розширення. Видаляти всі заглушки не слід, тому що через отвори, що не використовуються, буде набиратися пил і з'явиться можливість попадання сторонніх предметів усередину системного блоку.

Завдання 6. Підключення органів управління

На лицьовій стороні корпусу системного блоку розташовано кілька кнопок керування (включення живлення, скидання) та світлодіодних індикаторів (живлення, робота накопичувачів). Для їх підключення на материнській платі є контактна панель, до якої приєднуються всі роз'єми, що йдуть від електронних елементів лицьової панелі корпусу, у тому числі від динаміка. Як правило, контактна панель розташована в одному з кутів материнської плати.

Підключати роз'єми слід відповідно до вказівок у документації на материнську плату. На материнській платі призначення груп контактів зазвичай маркується фарбою. Для визначення призначення штекерів знайдіть на їх корпусах умовні позначення, а якщо вони незрозумілі або написів немає, то простежте, куди йде провід від того чи іншого роз'єму. При підключенні світлодіодних індикаторів зверніть увагу на полярність підключення гнізда. Зазвичай кольоровий проводок підключається до "+", а чорний або білий - до "-" на материнській платі (роз'єм "+" маркується фарбою).

Завдання 7. Підключення роз'ємів живлення

Блок живлення завжди закріплений на корпусі системного блоку і в процесі збирання, як правило, не знімається. Винятком можуть бути випадки, коли використовується малогабаритний корпус та блок живлення заважає установці материнської плати.

Насамперед слід підключити роз'єм кулера (вентилятора системи охолодження процесора). Зазвичай роз'єм живлення кулера представлений у вигляді мініатюрного триконтактного роз'єму, який слід приєднати до аналогічного роз'єму на материнській платі.

Потім підключається основний роз'єм блоку живлення. Роз'єм на кабелі повинен до кінця піти в частину у відповідь на материнській платі, а клямка роз'єму надійно зафіксуватися. У ряді випадків для живлення процесора використовується додатковий роз'єм +12 В. Необхідність такого варіанта треба уточнити за документацією, чотириконтактний роз'єм для цього ланцюга завжди знаходиться біля сокету процесора.

Завдання 8. Встановлення накопичувачів

Встановлення накопичувача на гнучких магнітних дисках, жорсткого диска, а також приводу CD/DVD-дисків здійснюється за одними і тими самими правилами.

Насамперед, накопичувачі потрібно встановити у відповідні їх фізичним розмірам відсіки корпусу системного блоку та надійно закріпити гвинтами. Після цього необхідно підключити їх до відповідних інтерфейсних роз'ємів та роз'ємів живлення.

Для підключення жорстких дисків на системній платі встановлюються один або два 40-контактні роз'єми для IDE-інтерфейсу або кілька SATA-роз'ємів. Для всіх сучасних IDE-вінчестерів потрібно використовувати 80-жильний шлейф. При використанні 40-жильного шлейфу вінчестер не зможе передавати інформацію з максимально доступною для нього швидкістю.

При установці жорсткого диска, перш за все, необхідно за допомогою перемичок, розташованих на задній стороні корпусу, встановити режим його роботи: Master або Slave (режим Cable використовується дуже рідко). Для системного жорсткого диска слід встановлювати режим Master. Далі до жорсткого диска підключаються інтерфейсний шлейф та роз'єм живлення. Практично завжди перший провід (червоний або чорний) шлейфа знаходиться ближче до гнізда живлення жорсткого диска.

Системні плати, що підтримують інтерфейс SATA, мають додаткові роз'єми для інтерфейсу SATA. До кожного такого роз'єму підключається лише один пристрій за допомогою кабелю SATA. Для підключення живлення використовується перехідник, який одним кінцем приєднується до гнізда живлення жорсткого диска SATA, а іншим – до гнізда на кабелі блоку живлення. Вінчестери з інтерфейсом SATA джамперів для вибору типу пристрою не мають.

Привід CD/DVD-дисків підключається так само, як і жорсткий диск. Але для підвищення продуктивності комп'ютера бажано підключати привод CD/DVD-дисків на другий канал (роз'єм) IDE-інтерфейсу. Для приводу CD-RW найкраще встановлювати режим Master. При установці двох оптичних приводів, наприклад CD-RW та DVD, їх можна підключити на один шлейф. Один привід повинен працювати у режимі Master, а другий – у Slave.

На приводі CD/DVD-дисків є додаткові роз'єми, призначені для підключення аналогового аудіокабелю до звукової плати. Цей кабель можна не встановлювати, якщо не прослуховувати через комп'ютер диски типу аудіо-CD.

Завдання 9. Встановлення плат розширення

Залежно від типу материнської плати на ній можуть бути встановлені слоти різних інтерфейсів: ISA, PCI, AGP, PCI Express. Тому перед встановленням на материнську плату потрібно уважно розглянути плати розширення та визначити, в які слоти вони можуть бути встановлені. Після того, як відповідність плат розширення та слотів встановлено, можна переходити до збирання.

При установці плати розширення слід акуратно ввести роз'єм плати в слот, а потім з невеликим зусиллям вставити роз'єм в слот до упору. Плату розширення необхідно зафіксувати, інакше в процесі експлуатації комп'ютера друкована плата рано чи пізно вилізе зі слота. Кріпиться плата за допомогою гвинта, що притягує задню металеву планку плати розширення до корпусу системного блоку.

Для надійної фіксації відеоконтролерів у слотах розширення AGP та PCI Express (формату 16x) встановлюють спеціальні тримачі із клямкою. У цьому випадку, коли ви встановлюєте відеокарту в слот, потрібно спочатку відігнути фіксатор у сторону, встановити плату, а потім відпустити фіксатор. При цьому можна почути характерне клацання, що свідчить про правильне встановлення та закріплення плати.

Завдання 10. Перевірка працездатності комп'ютера

Після складання системного блоку, не закриваючи корпуса комп'ютера, підключіть до нього клавіатуру, монітор та мишу.

Інші зовнішні пристрої для першого увімкнення комп'ютера не слід підключати.

При включенні живлення слід звернути увагу на звукові сигнали, які видає внутрішній динамік, та на світлодіоди, розташовані на системному блоці та клавіатурі.

На системному блоці має засвітитися світлодіод, який індикує перехід комп'ютера до робочого стану. Одночасно повинні запрацювати всі вентилятори системного блоку: у блоці живлення, кулери процесора, відеокарти та чіпсета (якщо вони там є). На деяких моделях системних плат вентилятор процесора вмикається не відразу, а через кілька секунд. Якщо

вентилятори не запрацювали, знову перевірте правильність складання системного блоку. Зверніть увагу на коректність підключення роз'ємів живлення та кнопки "Мережа" (Power SW).

Якщо після увімкнення живлення внутрішній динамік видає тривалий або багаторазові сигнали, то по таблиці звукових сигналів BIOS (див. посібник до материнської плати) слід визначити несправний вузол. У версії BIOS, розроблених різними фірмами, таблиці сигналів досить сильно відрізняються. Визначивши несправність, слід усунути її шляхом виправлення допущених під час складання помилок або заміни несправних вузлів.

2.3. Зміст звіту

Перелік та короткий опис компонентів системного блоку робочого комп'ютера студента.

Перелік та короткий опис компонентів системного блоку комп'ютера, про який мріє студент.

Письмові відповіді на контрольні питання.

Висновки.

2.4. Контрольні питання

1. Які основні блоки та плати входять до складу персонального комп'ютера?
2. Яка стандартна потужність блоку живлення? Які напруги видає блок живлення? У чому основні причини виходу з ладу блоку живлення?
3. Які типи корпусів застосовують у ПК?
4. Які основні пристрої знаходяться на материнській платі?
5. Для чого служить постійний пристрій?
6. Якими основними параметрами характеризуються накопичувачі на жорстких дисках?
7. Які сучасні типи інтерфейсів використовуються для НЖМД?
8. Що таке шина на системній платі? Які типи шин ви знаєте?
9. Навіщо потрібна кеш-пам'ять? На яких мікросхемах вона реалізується?
10. Які типи модулів ОЗП застосовують у сучасних ПК?
11. Які основні параметри CD-ROM-приводів?
12. Які основні характеристики процесорів?
13. Які види накопичувачів на лазерних дисках ви знаєте?
14. Які види дискових накопичувачів ви знаєте?
15. Що таке порт? Які типи портів ви знаєте?
16. Структура жорсткого диска.
17. Які види пам'яті ви знаєте?
18. Що таке BIOS? Основні функції BIOS.
19. Які основні характеристики системної плати можна назвати?
20. Який пристрій використовується для виведення графічної інформації?
21. Які основні параметри BD-ROM-приводів?

РОБОТА №3. ВСТУП ДО СИСТЕМИ МОДЕЛЮВАННЯ ЕЛЕКТРОННИХ СХЕМ MULTISIM

Мета роботи: Вивчення системи меню, складу бібліотек компонентів і набору програм, що імітують базові елементи комп'ютерних систем, а також освоєння навичок роботи із складання електронних схем в програмі Multisim.

Програмне забезпечення: online версія програми Multisim (https://www.multisim.com/).

У процесі виконання роботи необхідно:

Ознайомитися з характером виконання команд меню програми Multisim.

– Вивчити склад моделей елементів різних каталогів бібліотеки

компонентів, які використовуються для складання принципових схем.

– Ознайомитися з групою індикаторних пристроїв і набором контрольно-вимірювальних приладів.
– Освоїти техніку складання принципових електричних схем комп'ютерних пристроїв і дослідження їх функціонування на прикладі операційного підсилювача.

3.1. Короткі теоретичні відомості

Програма Multisim [9, с. 92] розроблена фірмою National Instruments Electronics Workbench Group і використовується для схемотехнічного моделювання аналогових та цифрових радіоелектронних пристроїв різного призначення та дає змогу простими засобами будувати на екрані монітора аналогову або цифрову електронну схему, під'єднувати електро- та радіовимірювальні прилади та отримувати результати або в вигляді цифрових даних на моніторах вимірювальних приладів або на логічному аналізаторі. Програма Multisim подібна до програми Electronics Workbench, випущеною тим же самим розробником. Лабораторний практикум, розроблений А.Л. Литвиновим [10], використовується в роботах №№ 3-8 даного посібника.

Перш ніж створювати креслення принципової схеми засобами програми Multisim, необхідно на аркуші паперу підготувати її ескіз із зразком розташування компонентів. У загальному випадку процес створення схеми починається з розміщення на робочому столі Multisim компонентів із бібліотек програми відповідно до підготовленого ескізу. Для відкриття каталогу потрібної бібліотеки необхідно підвести курсор миші до відповідної ікони і натиснути один раз її ліву кнопку. На рис 3.1 відкритий каталог Джерела. Каталог у програмі ще називають групою. Необхідний для створення схеми символ компоненту переноситься з каталогу на робоче поле програми рухом миші при натиснутій лівій кнопці, після чого кнопка відпускається (для фіксації символу) і проводиться подвійне клацання по значку компоненту. У панелі конфігупації, що розкривається з правого боку, встановлюються необхідні параметри (опори резисторів, тип транзисторів, ємності конденсаторів і т. п.). На рис. 3.1 встановлена напруга джерела постіного току 12 В. Вибір підтверджується натисненням лівої кнопки миші. На цьому етапі також необхідно передбачити місце для розміщення контрольних крапок і ікон контрольно-вимірювальних приладів.

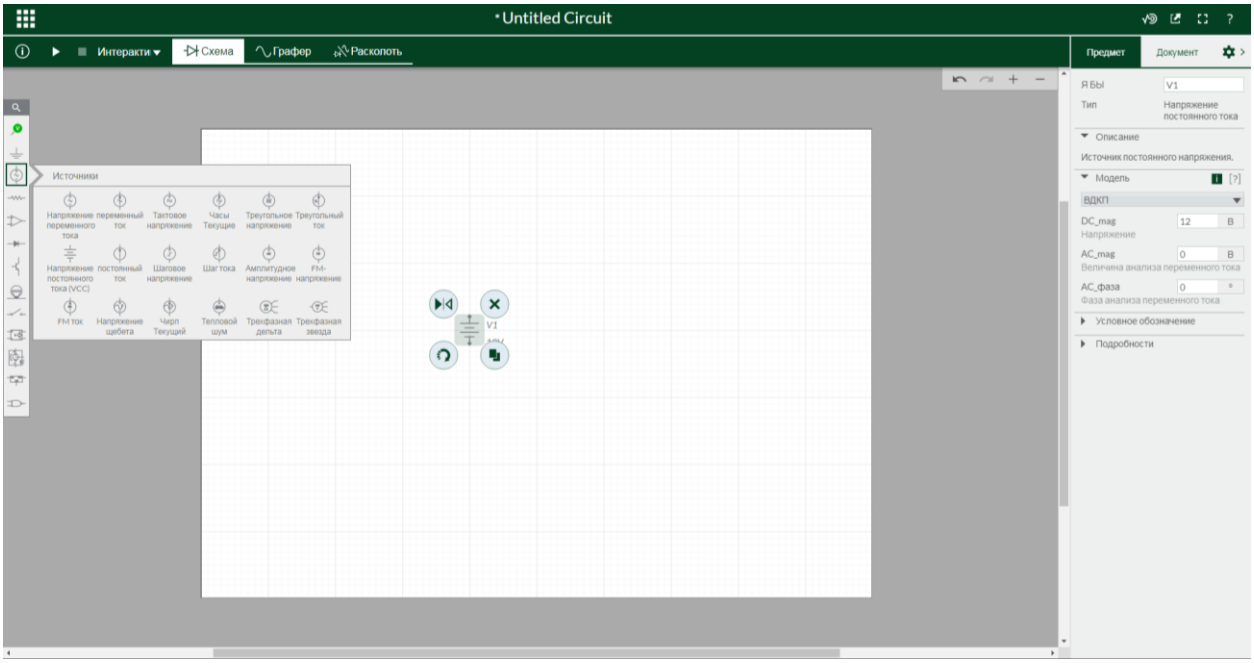


Рисунок 3.1 – Зовнішній вигляд робочого столу програми Multisim

Після розміщення компонентів проводиться з’єднання їх виводів провідниками. Для виконання під’єднання курсор миші підводиться до виведення компоненту і після появи зображення котушки з провідником натискається ліва кнопка і провідник, що з’являється, протягується до виведення іншого компоненту, після чого натискається ліва кнопка миші, і з’єднання готове. Якщо результат виконаної операції користувача не задовольняє можна видалити провідник, натиснув символ «хрестик» та повторити операцію. Якщо все вийшло як слід, то натискається ліва кнопка миші, що фіксує побудований провідник. У разі необхідності під’єднання до створених провідників інших провідників у каталозі «Схема роз'ємів» вибирається крапка (● – символ з’єднання) і переноситься на раніше встановлений провідник. Якщо ця крапка дійсно має електричне з’єднання з провідником, то вона забарвлюється червоним кольором. Після установки до точки з’єднання можна під’єднати ще два провідники. Якщо з’єднання потрібно розірвати, то курсор підводиться до одного з провідників та натискається ліва кнопка миші, після чого треба клацнути по символу «хрестик». Якщо необхідно під’єднати вивід до провідника, що є на схемі, то провідник від виведення компоненту курсором підводиться до вказаного провідника і після появи точки з’єднання та натискається ліва кнопка миші. Прокладання сполучних провідників проводиться автоматично, а перешкоди – компоненти і інші провідники огинаються по ортогональних напрямах.

Для переміщення окремого сегмента провідника до нього підводиться курсор, натискається ліва кнопка і після появи у вертикальній або горизонтальній площині подвійного курсора проводяться потрібні переміщення.

Під’єднання до схеми контрольно-вимірювальних приладів проводиться аналогічно. Після підготовки схеми рекомендується скласти її опис. Інформаційна панель позначена літерою і, обведеною колом. Після натиснення лівою кнопкою миші на цю іконку з лівого боку виникає інформаційна панель, де вказується ім’я схеми та її призначення.

3.2. Порядок виконання роботи

1. Помістить у вікно програми Multisim зображення джерела постійної напруги з групи Джерела. Відпрацюйте на ньому команди Rotate, Flip/Miror

Duplicate, Delete. Клацніть лівою кнопкою миші по елементу джерела постійної напруги, щоб відкрити панель конфігурації. Панелі конфігурації відповідає кнопка на панелі інструментів. Тут можна задавати різні значення напруги (DC_mag). Помістить у вікно вольтметр (група Аналіз і анотація –> Напруга) і під'єднайте його до джерела напруги. Запустіть схему і порівняйте виміряну напругу і напругу на джерелі. Під'єднайте до схеми елемент «земля» (група Схема роз'ємів).

2. Зберіть у системі Multisim схему за рисунком 3.2, яка буде об'єктом досліджень. Схему занесіть у звіт.

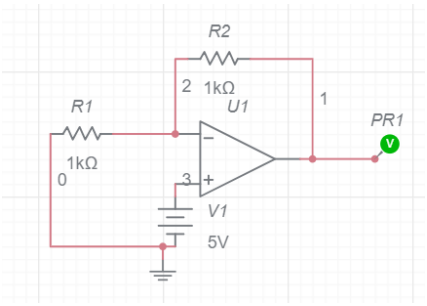


Рисунок 3.2 – Принципова схема операційного підсилювача

3. Натиснувши на кнопку відкрийте панель конфігурації (рис. 3.2) та в настройках моделювання виберіть один з наступних типів моделювання :

Інтерактивне моделювання — Використовуйте для імітації осцилографа. Виконує моделювання в часовій області (перехідне) і працює доти, доки не буде призупинено або зупинене.

Моделювання перехідних процесів – використовується для вимірювання відгуку схеми залежно від часу. Зупиняється через заданий час.

Розгортка змінного струму — використовується для вимірювання частотної характеристики лінійного ланцюга. Для конструкцій із компонентами змінного струму, наприклад, джерело змінної напруги.

Робоча точка постійного струму використовується для вимірювання робочої точки постійного струму, також відомої як точка зміщення ланцюга. Це напруга або струм, що встановилася, на зазначеному виводі активного пристрою без подачі вхідного сигналу.

Параметри моделювання змінюються залежно від вибраного типу моделювання. Параметри симулятора застосовуються до всіх типів моделювання. Стандартні значення параметрів зазвичай прийнятні та змінювати їх непотрібно.

Для нашої схеми вибираємо *Моделювання перехідних процесів* та запускаємо режим симуляції з відкритої панелі конфігурації, натиснувши на рядок *Розпочати симуляцію*. Це також можна зробити з панелі інструментів, натиснувши кнопку, під якою вспливає підказка *Run simulation (Ctrl+R).*

Після цього можна подивитися на результат моделювання, звернувшись до пункту меню *Графер*. Вигляд результату буде залежати від виду моделювання та наявності контрольних приладів у схемі. В нашему випадку ми побачимо пряму горизонтальну лінію що відповідає значенню напруги у 12 В (рис. 3.3). Тобто на виході вона така ж сама як і на вході, що спостерігається не завжди і залежить від значення вхідної напруги та опорів схеми – R_1 і R_2.

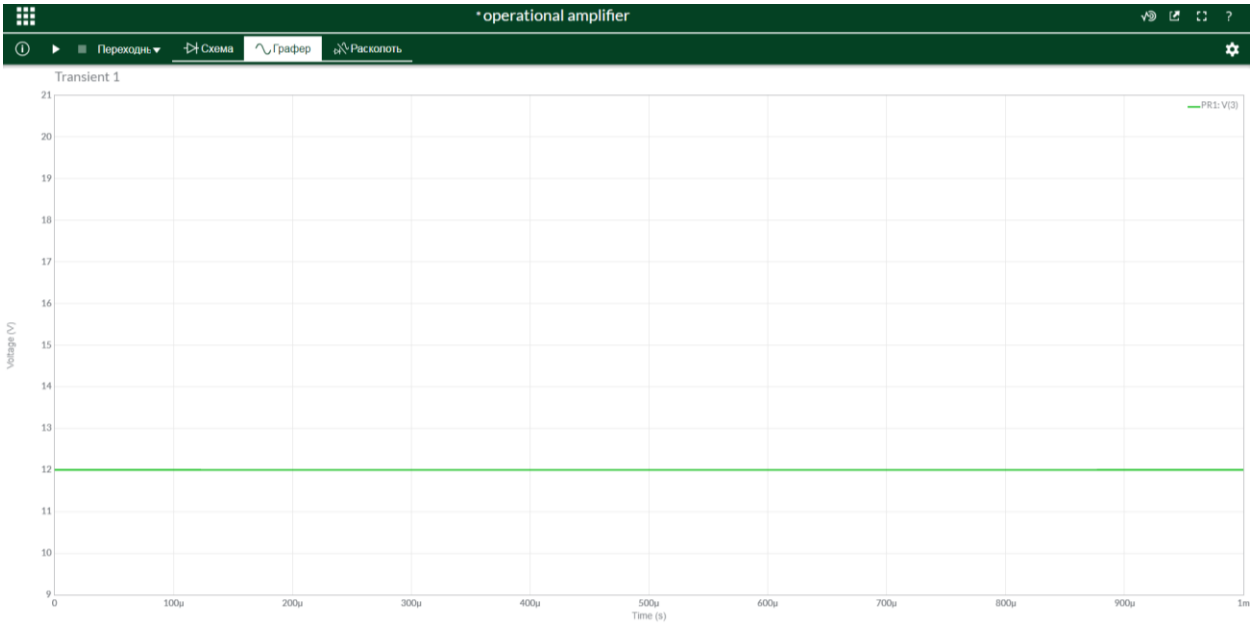


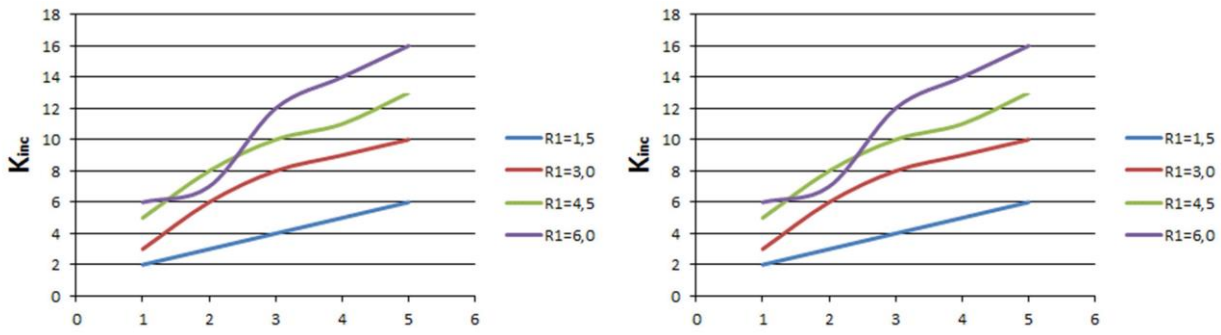
Рисунок 3.3 – Графік зміни виходної напруги у часі

4. Скопіюйте в звіт таблицю 3.1. Встановіть напругу джерела U_{in} = № варіанта/10. Проведіть над об'єктом двофакторний експеримент, у якому як вхідні фізичні фактори будуть використовуватися значення опорів схеми – R_1 і R_2, а відгук – коефіцієнт підсилення $K_{inc} = U_{out} / U_{in}$. U_{out} – напруга на виході операційного підсилювача (В). Результати занесіть у таблицю 3.1. Теоретичне значення коефіцієнта підсилення підрахуйте за формулою $K_{inc} = 1 + R_2/R_1$ і занесіть в таблицю

Таблиця 3.1 – Значення коефіцієнта підсилення (K_{inc})

R1 (кОм)	R2 (кОм)					Теорія
	1,0	2,0	3,0	4,0	5,0	$K_{inc} = 1 + R_2/R_1$
1,5						
3,0						
4,5						
6,0						

За таблицею 3.1 побудуйте в Excel графіки залежності коефіцієнта підсилення операційного підсилювача K_{inc} від R_1 і R_2 при постійному U . За зразок візьміть рисунок 3.4.



R1 R2

Рисунок 3.4 – Графіки залежності коефіцієнта підсилення операційного підсилювача K_{inc} від R_1 і R_2

Сформулюйте висновки по експерименту.

3.3. Зміст звіту

Звіт повинен містити: принципову схему досліджуваного пристрою , таблицю значень коефіцієнта підсилювання K_{inc}, графіки залежності коефіцієнта підсилення від R_1 і R_2, основні характеристики підсилювального каскаду відповідно до завдання, письмові відповіді на контрольні питання, висновки.

3.4. Контрольні питання

1. Яка структура вікна Multisim?
2. Що потрібно зробити перш ніж створювати креслення принципової схеми засобами програми Multisim?
3. Як перенести на робочий стіл Multisim електричні компоненти?
4. Як з’єднати між собою компоненти електричної схеми?
5. Як встановити параметри електричного елемента?
6. Як можна виділити об’єкт на робочому столі і що з ним після цього можна зробити?

РОБОТА №4. ДОСЛІДЖЕННЯ ФУНКЦІОНУВАННЯ БЛОКА ЖИВЛЕННЯ СТАЦІОНАРНИХ КОМП'ЮТЕРІВ

Мета роботи: знайомство з принципом побудови блока живлення стаціонарних комп'ютерів, технологією його функціонування і засобами перетворення змінної напруги електричної мережі в постійну для живлення блоків комп'ютера.

У процесі виконання роботи необхідно:

Ознайомитися з принципом побудови і функціонування напівпровідникового діода, а також з типовими схемами його використання для перетворення змінної напруги в постійну.

Ознайомитися з схемами побудови лінійного і імпульсного блока живлення, порівняти їхні характеристики.

Дослідити вплив параметрів блока живлення (схема випрямляча та ємкість його конденсатора) на вихідну напругу.

4.1 Теоретичні відомості

Блок живлення стаціонарного комп'ютера – це вторинне джерело живлення, призначене для забезпечення вузлів комп'ютера електричною енергією постійного струму. У його завдання входить перетворення мережевої напруги до заданих значень напруги живлення, її стабілізація і захист від незначних перешкод з боку електричних мереж живлення. Завдяки вбудованому вентилятору, він також бере участь в охолоджуванні системного блока. У блоку живлення для перетворення змінного струму в постійний використовуються напівпровідникові діоди.

Напівпровідниковий діод – це найпростіший напівпровідниковий прилад, що складається з одного PN-переходу. Основна його функція – проводити електричний струм в одному напрямку, і не пропускати його в зворотному. Складається діод із двох шарів напівпровідника типів N і P, як показано на рисунку 4.1. Позначення діода на електричних схемах приведено на рисунку 4.2.

Рисунок 4.1 – Будова діода

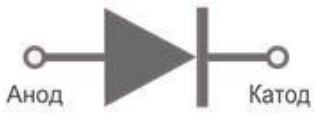


Рисунок 4.2 – Позначення діода на схемі

Якщо ні до анода, а ні до катода не підімкнена напруга, то діод знаходиться в стані спокою. У частині N є в наявності вільні електрони – негативно заряджені частинки. У частині P знаходяться позитивно заряджені іони – дірки. У результаті у тому місці, де є частки з зарядами різних знаків, виникає електричне поле, яке притягує їх один до одного.

Під дією цього поля вільні електрони з частини N дрейфують через PN перехід у частину P і заповнюють деякі дірки. У підсумку виходить дуже слабкий електричний струм, вимірюваний в наноамперах. У результаті щільність речовини в P частини підвищується і виникає дифузія (прагнення речовини до рівномірної концентрації), що штовхає частки назад на бік N.

Якщо підімкнути плюс джерела живлення до катода, а мінус – до анода, то відповідно до сили тяжіння, яка виникла між зарядами різної полярності, електрони з області N почнуть рух до плюса і віддаляться від PN-переходу. Аналогічно, дірки з P будуть притягатися до мінуса, і також віддаляться від PN-переходу. Отже, щільність речовини у електродів підвищується. У дію приходить дифузія і починає штовхати частки назад, прагнучи до рівномірної щільності речовини. У результаті діод не проводитиме струм.

Якщо підімкнути плюс джерела живлення до анода, а мінус – до катода, то між зарядами однаковою полярності виникає сила відштовхування. Негативно заряджені електрони віддаляються від мінуса і рухаються у бік PN-переходу. Зі свого боку, позитивно заряджені дірки відштовхуються від плюса і направляються назустріч електронам. PN-перехід збагачується зарядженими частинками з різною полярністю, між якими виникає електричне поле – внутрішнє електричне поле PN-переходу. Під його дією електрони починають дрейфувати на бік P. Частина з них рекомбінують із дірками (заповнюють місце в атомах, де не вистачає електрона). Решта електронів спрямовуються до плюса джерела живлення. Через діод пішов струм. Ця властивість функціонування діода використовується для побудови джерела живлення стаціонарних комп'ютерів.

Основне завдання блока живлення (рис. 4.3) – це перетворення змінної напруги мережі 220 В, 50 Гц, в напругу живлення конструктивних елементів комп'ютера ± 12 В, ± 5 В, ± 3,3 В [10].

Рисунок 4.3 – Зовнішній вигляд блока живлення комп'ютера

Раніше для цього застосовувалися лінійні блоки живлення на основі силових трансформаторів, вагою до 6 кг. Сучасні блоки живлення будуються за імпульсною схемою і важать до 1 кг. Структурна схема імпульсного блока живлення для перетворення змінної напруги 220 В в постійну 12 В зображена на рисунку 4.4.

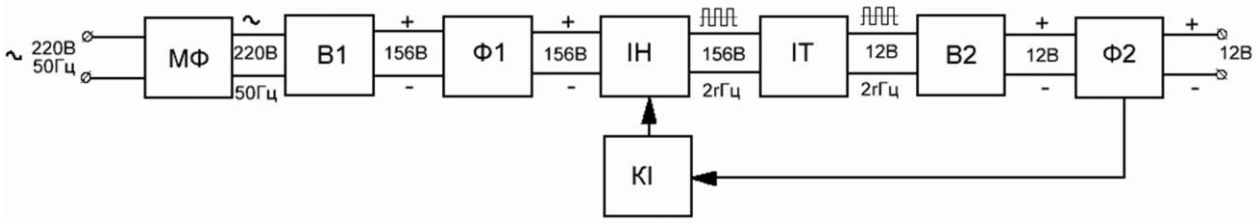


Рисунок 4.4 – Структурна схема імпульсного блока живлення комп'ютера

Позначення на схемі:

МФ – мережевий фільтр, його завдання мінімізувати мережеві перешкоди, як вхідні, так і вихідні, що виникають унаслідок роботи блока живлення;

В1 – перший випрямляч напруги, побудований на діодах. Його завдання – перетворити змінну напругу в пульсуючу, але однієї полярності, в якій присутня постійна складова;

Ф1 – перший згладжувальний фільтр. Його завдання – згладити пульсації, щоб отримати справжню напругу постійного струму;

ІН – інвертор напруги. Він здійснює формування імпульсної прямокутної напруги високої частоти;

ІТ – імпульсний трансформатор. Його завдання – знизити напругу до потрібної величини і здійснити гальванічну розв’язку ланцюгів. Завдяки використанню імпульсної прямокутної напруги високої частоти його габарити і вага значно менше, ніж у трансформатора в лінійних ланцюгах;

В2 – другий випрямляч напруги, побудований на діодах;

Ф2 – другий згладжувальний фільтр;

КІ – контролер інвертора. Його завдання – управління інвертором, завдяки зворотному зв’язку здійснювати стабілізацію напруги на виході блока живлення.

Для формування постійних напруг у 3,3 В, 5 В використовуються декілька вторинних обмоток імпульсного трансформатора.

4.2 Порядок виконання роботи

А. Дослідження вхідної напруги.

Зберіть схему за рисунком 4.5.

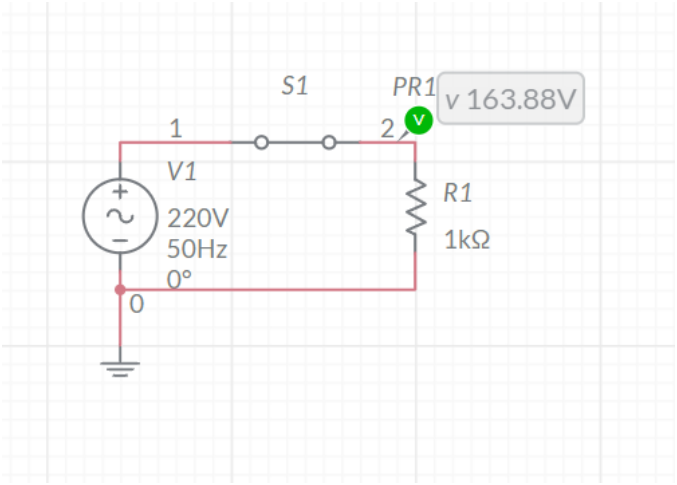


Рисунок 4.5 – Схема для дослідження вхідної напруги

Налаштувавши осцилограф, скопіюйте в звіт за допомогою інструмента «Ножницы» операційної системи Windows графік зміни напруги, вказавши частоту і величину напруги.

Б. Дослідження випрямляча напруги.

Зберіть схему за рисунком 4.6. Це так званий однопівперіодний випрямляч.

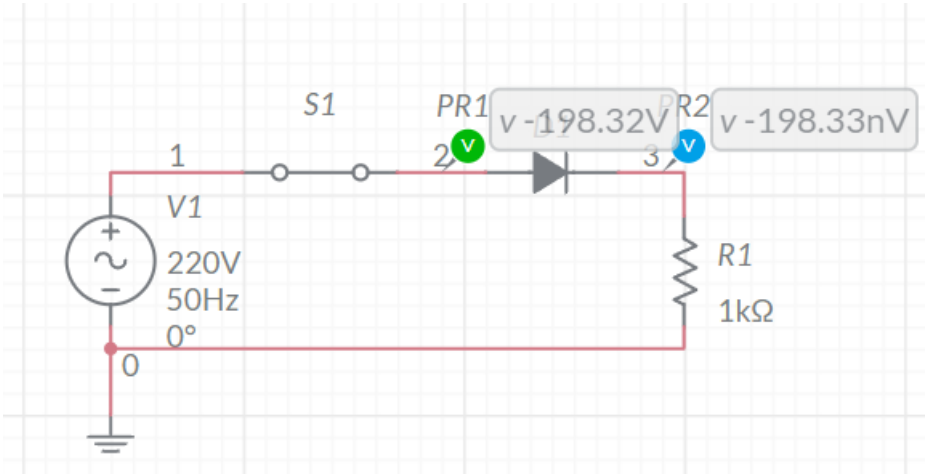


Рисунок 4.6 – Схема для дослідження однопівперіодного випрямляча

Налаштувавши осцилограф, скопіюйте у таблицю 4.1 за допомогою інструмента «Ножиці» операційної системи Windows графіки зміни напруги (епюра напруги) на вході і виході схеми. Поясніть зміни на графіку.

В. Дослідження впливу конденсатора на процес.

Зберіть схему за рисунком 4.7.

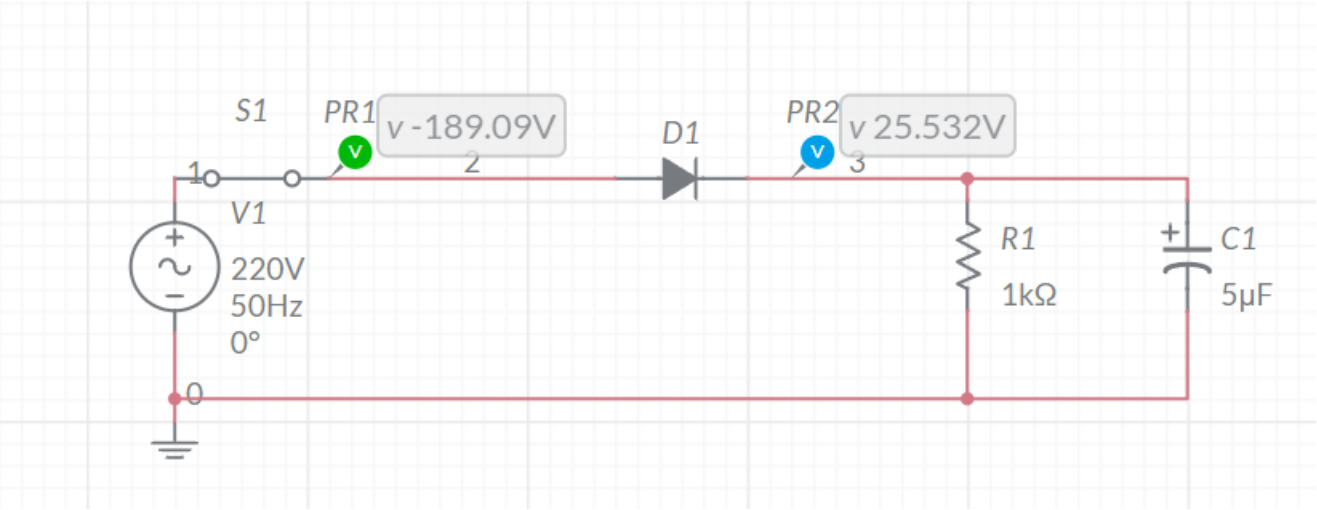


Рисунок 4.7 – Схема для дослідження впливу конденсатора на процес

Скопіюйте в таблицю 4.1 графік напруги на виході схеми при значення конденсатора 5 µF. Змінюючи ємкість конденсатора, скопіюйте в таблицю 4.1 відповідні епюри. Зробить висновок про вплив ємкості конденсатора на вихідну напругу.

Таблиця 4.1 – Залежність епюр напруг від ємкості конденсатора

Епюра напруги ⇒ на вході схеми	
Ємкість конденсато-ру	Епюра напруги на виході схеми
Конденсатор відсут-ній	
5 µF	
20 µF	
50 µF	
100 µF	

Г. Дослідження двопівперіодного випрямляча.

Розглянуті схеми випрямлення струму використовують тільки додатну складову вихідного змінного струму, тобто 50 % енергії втрачається. Цей недолік усувається за допомогою мостової схеми, яка містить 4 діода, увімкнених у певний спосіб. Зберіть схему за рисунком 4.8.

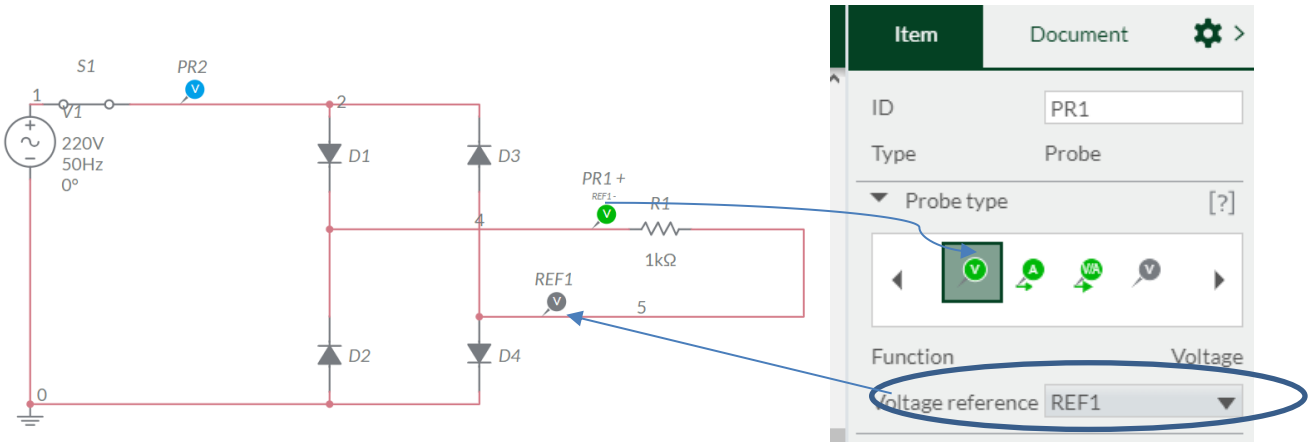


Рисунок 4.8 – Схема для дослідження мостової схеми випрямляча

Налаштувавши осцилограф, скопіюйте в таблицю 4.2 графік зміни напруги на вході і виході схеми. Поясніть зміни в графіку порівняно зі схемою на рисунку 4.7. Розберіть, як працює мостова схема випрямляча

Таблиця 4.2 – Залежність епюр напруг від ємкості конденсатора при мостовій схемі випрямляча

Ємність конденсатора	Епюра (графік) напруги на виході схеми
0 мкФ (відсутній)	
5 мкФ	
20 мкФ	
50 мкФ	
100 мкФ	

Зберіть схему за рисунком 4.9:

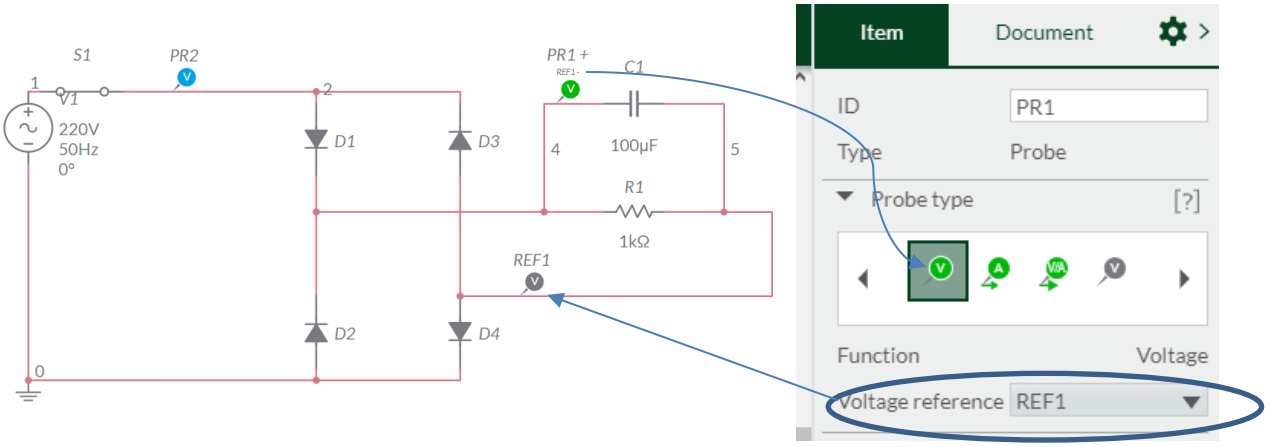


Рисунок 4.9 – Схема для дослідження мостової схеми випрямляча з конденсатором

Скопіюйте в таблицю 4.2 графік напруги на виході схеми при значенні конденсатора 5 μF. Змінюючи ємкість конденсатора, скопіюйте в таблицю 4.2 відповідні епюри. Зробить висновок про вплив ємкості конденсатора на вихідну напругу. Порівняйте епюри зміни напруги в таблицях 4.1 і 4.2 за однакових значеннях конденсатора. Зробить висновки за цими порівняннями.

Контрольні запитання:

1. Яка роль джерела живлення в комп'ютері?
2. Які види джерел живлення використовуються в техніці?
3. Як працює імпульсне джерело живлення і які він має переваги перед лінійним джерелом живлення?
4. У чому полягає особливість електропровідності напівпровідників?
5. Яке призначення напівпровідникових діодів?
6. Які схеми випрямлячів використовують у техніці ?
7. Як згладжуються пульсації току в випрямлячах?
8. Яке призначення імпульсного трансформатора у блоці живлення?

РОБОТА №5. ДОСЛІДЖЕННЯ ЛОГІЧНИХ СХЕМ І СХЕМ ПАМ'ЯТІ НА ДІОДАХ І ТРАНЗИСТОРАХ

Мета роботи: знайомство з реалізацією базових логічних схем комп'ютерних систем і осередку пам'яті на діодах і транзисторах.

У процесі виконання роботи необхідно:

Ознайомитися з принципом побудови і функціонування схем І, НІ, АБО та інших на напівпровідникових діодах і транзисторах. Набути навички складання таблиць істинності і визначати логічну функцію по таблиці істинності.

Теоретичні відомості

Напівпровідниковий діод – це найпростіший напівпровідниковий прилад, який складається з двох шарів напівпровідника типів N і P. Вони утворюють PN-перехід. На рисунку 5.1 наведено позначення діода на електричних схемах.

Рисунок 5.1 – Позначення діода на електричних схемах

Якщо підімкнути плюс джерела живлення до аноду, а мінус – до катоду через резистор, то через діод піде струм, як показано на рисунку 5.2.

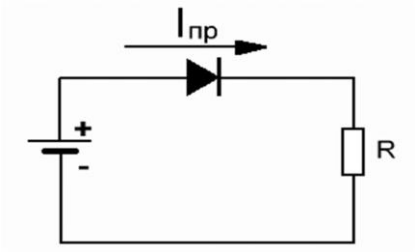
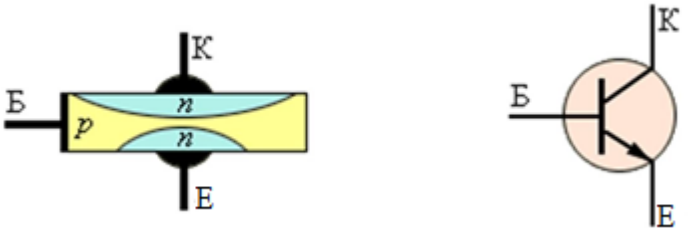


Рисунок 5.2 – Пряме під'єднання діода

Якщо під'єднати плюс джерела живлення до катоду, а мінус – до аноду через резистор, то через діод струм йти не буде. Цю властивість можна використовувати для побудови логічних схем.

Для реалізації осередків пам'яті і логічних схем у комп'ютерній техніці здебільшого використовують напівпровідникові транзистори.

Їх два типи: польові та біполярні. Розглянемо будову і роботу біполярного транзистора. Він складається з трьох областей: емітера, бази і колектора, на кожну з яких подається напруга. Залежно від типу провідності цих областей, виділяють n-p-n і p-n-p транзистори. На рисунку 5.3 зображено будову біполярного транзистора n-p-n типу і його умовне зображення на електричних схемах [4].

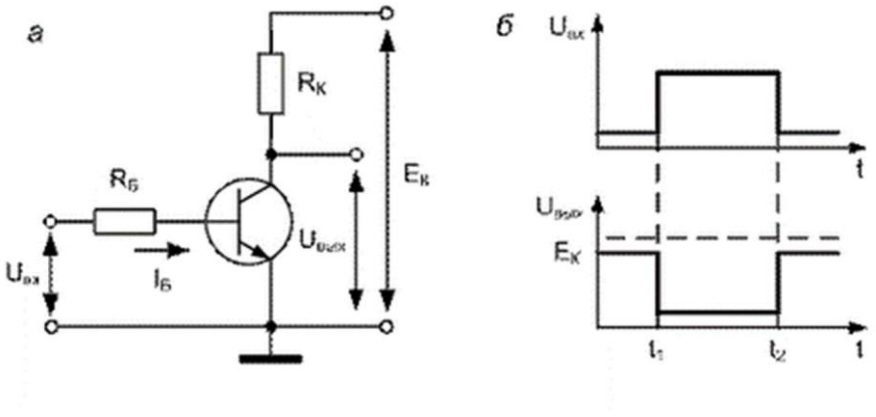


а б

Рисунок 5.3 – Напівпровідниковий транзистор:

а – будова напівпровідникового транзистора; б – умовне зображення транзистора на електричних схемах

Принцип роботи транзистора: між емітером і колектором тече сильний струм (струм колектора), а між емітером і базою – слабкий керівний струм (струм бази). Струм колектора буде змінюватися залежно від зміни струму бази. У комп'ютерних системах транзистори переважно працюють у режимі ключа. На рисунку 5.4 зображена типова електрична схема електронного ключа на біполярному n-p-n транзисторі і епюри напруги на його вході і виході.



а б

Рисунок 5.4 – Ключовий режим роботи транзистора:

а – електрична схема електронного ключа на біполярному n-p-n транзисторі; б – епюри напруги на його вході та виході схеми

Залежно від стану p-n переходів транзисторів розрізняють три види його роботи – режим відсічення, режим насичення і лінійний режим.

Якщо $U_{вх} \approx 0$, то обидва переходу транзистора (і емітерний, і колекторний) будуть закриті, струм бази у цьому випадку дорівнює нулю і транзистор переходить у режим відсічення. Опір по ланцюгу колектор-емітер буде максимальним і $U_{вих} \approx E_к$.

Якщо $U_{вх} = U_{max}$, то обидва перехода транзистора (і емітерний, і колекторний) будуть відкриті, у транзисторі відбувається вільний перехід носіїв зарядів, струм бази буде максимальний, струм колектора буде дорівнює струму колектора насичення і транзистор переходить у режим насичення. Опір по ланцюгу колектор-емітер буде мінімальним і $U_{вих} \approx 0$.

За $0 < U_{вх} < U_{max}$ транзистор працює в лінійному режимі як підсилювач.

Ключовий режим роботи транзистора дозволяє будувати на його основі як логічні елементи, так і осередки пам'яті.

Порядок виконання роботи

А. У Electronics Workbench зібрати схему за рисунком 5.5

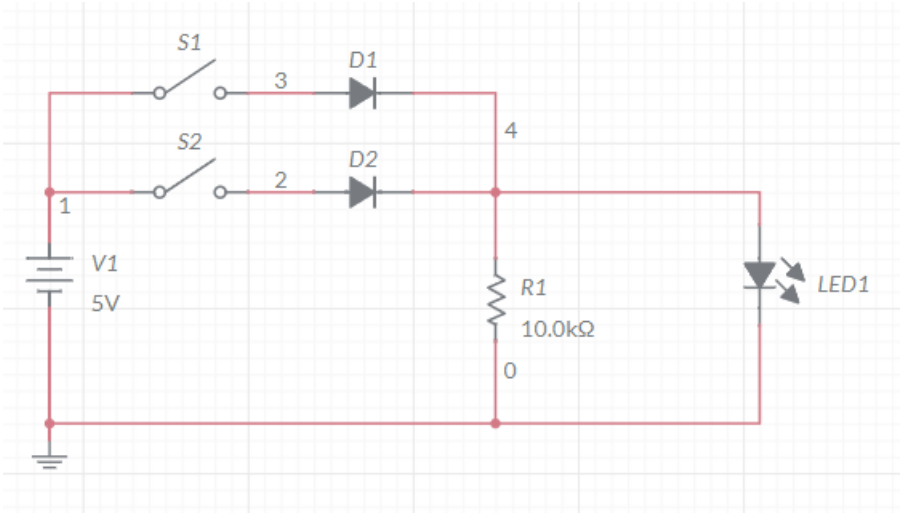


Рисунок 5.5 – Електрична схема до п. А

Перебравши всі можливі логічні комбінації ключів S1 і S2, заповніть таблицю істинності (табл. 3.1) для заданої схеми.

Примітка. Індикатор горить – логічна одиниця, не горить – логічний нуль.

Таблиця 5.1 – Таблиця істинності до схеми 3.5

S1	S2	Y
0	0	
0	1	
1	0	
1	1	

Визначте логічну функцію, яку реалізує схема на рисунку 5.5. Поясніть, як працює ця схема. Як вона зображується на логічних схемах?

Б. У Electronics Workbench зібрати схему за рисунком 5.6

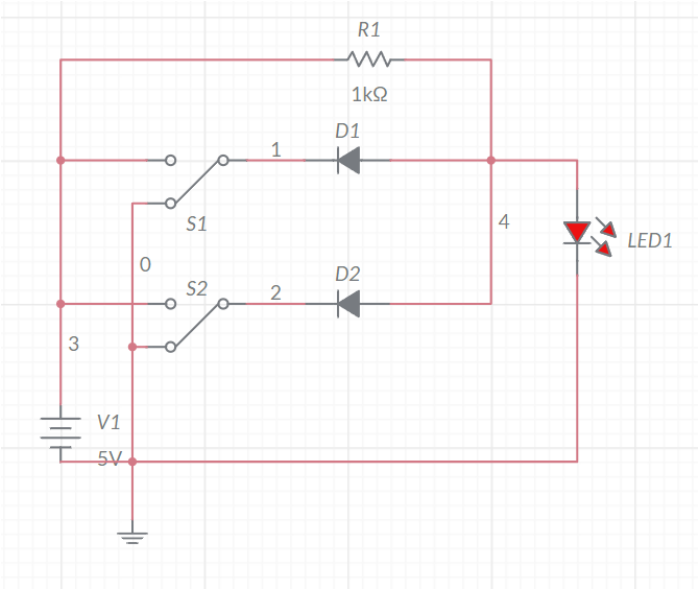


Рисунок 5.6 – Електрична схема до п. Б

Перебравши всі можливі логічні комбінації ключів S1 і S2, заповніть таблицю істинності (табл. 5.2) для заданої схеми.

Таблиця 5.2 – Таблиця істинності до схеми 5.6

S1	S2	Y (LED1)
0	0	
0	1	
1	0	
1	1	

Визначте логічну функцію, яку реалізує схема на рисунку 5.6. Поясніть, як працює ця схема. Як вона зображується на логічних схемах?

В. У Electronics Workbench зібрати схему за рисунком 5.7.

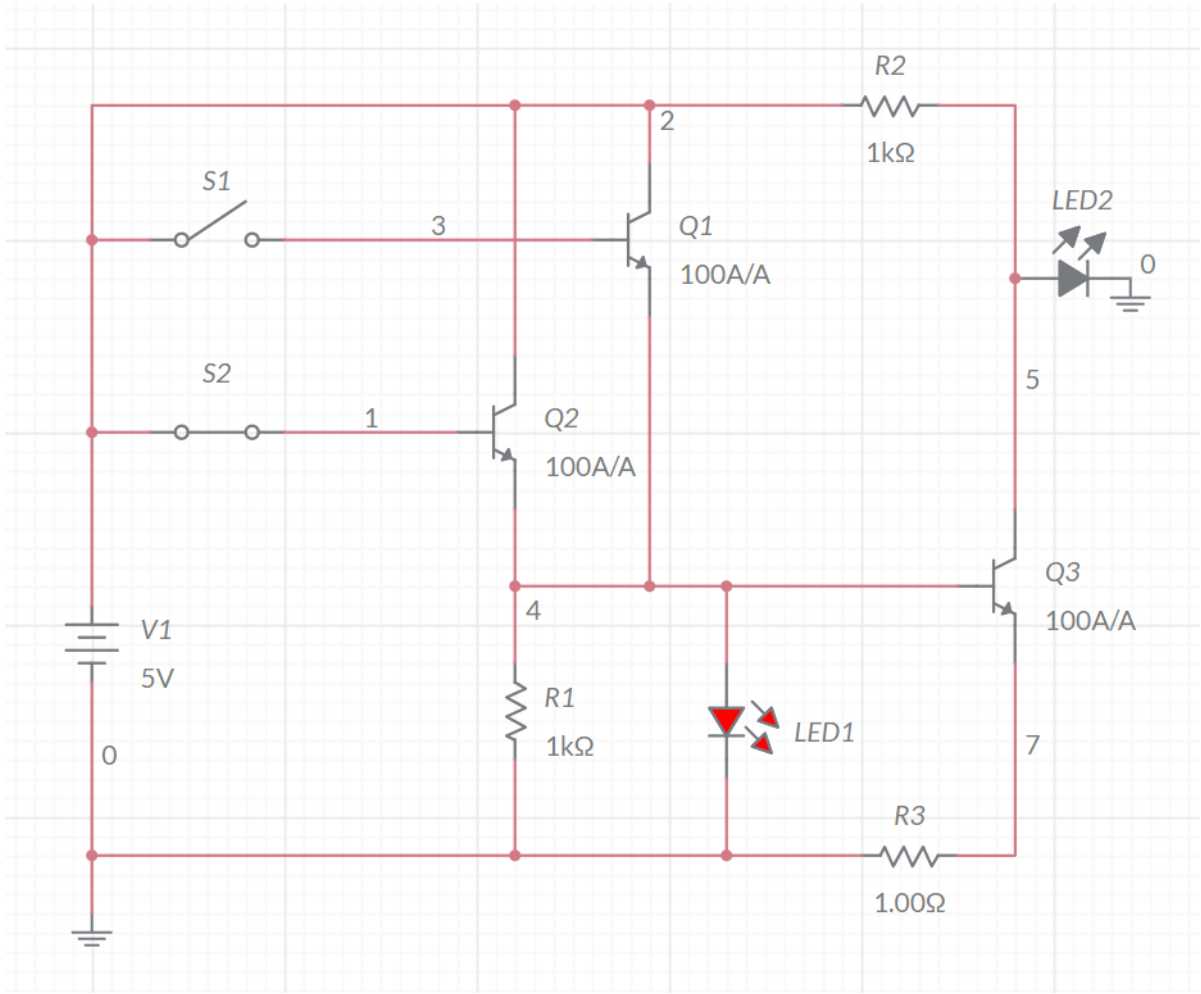


Рисунок 5.7 – Електрична схема до п. В

Перебравши всі можливі логічні комбінації ключів S1 і S2, заповніть таблицю істинності (табл. 3.3) для заданої схеми.

Таблиця 5.3 – Таблиця істинності до схеми 3.7

S1	S2	Y1 (LED1)	Y2 (LED2)
0	0		
0	1		
1	0		
1	1		

Визначте логічні функції, яку реалізує схема на рисунок 5.7. Поясність як працює схема. Як вона зображуються на логічних елементах?

Г. У Electronics Workbench зібрати схему за рисунком 5.8.

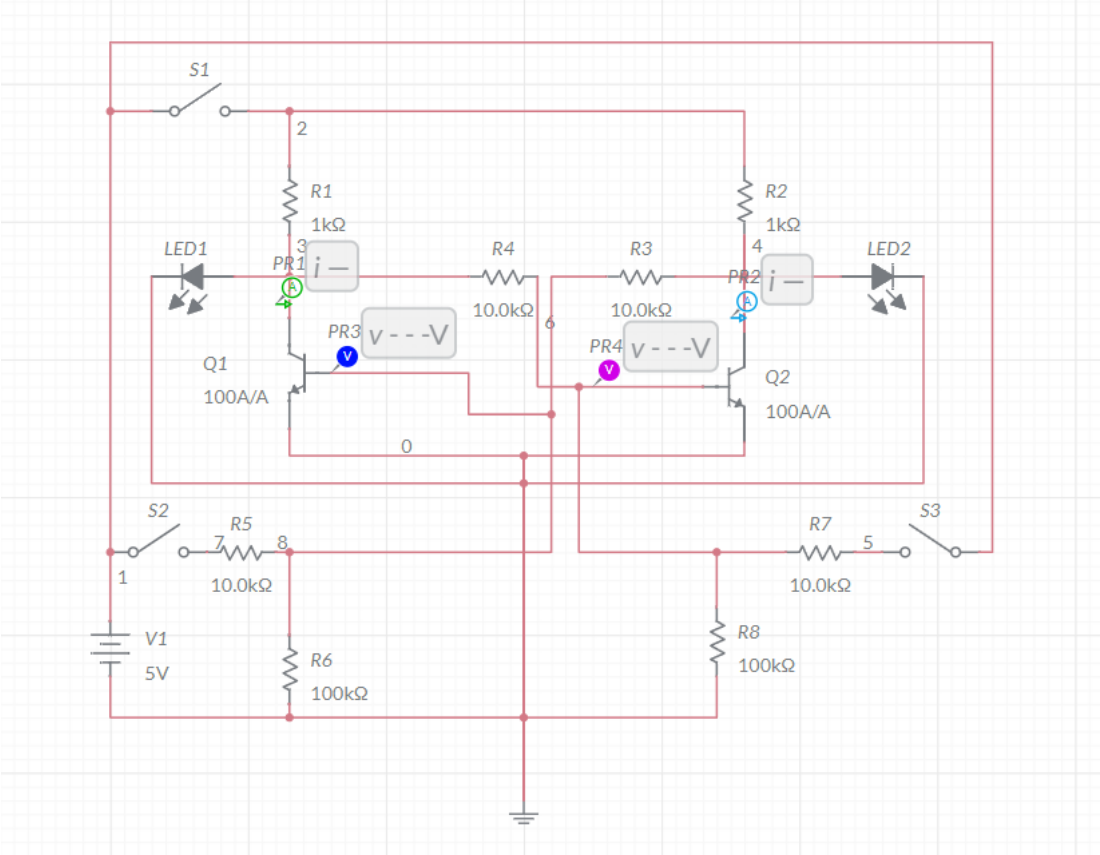


Рисунок 5.8 – Електрична схема до п. Г

Замкнувши S1, увімкніть схему в роботу. Відобразіть початковий стан схеми в першому рядку таблиці 5.4.

Примітка. Індикатор горить – логічна одиниця, не горить – логічний нуль.

Перебравши всі можливі логічні комбінації ключів S2 і S3, заповніть таблицю для схеми.

Таблиця 5.4 – Таблиця станів до схеми рисунка 5.8

S2	S3	LED1	LED2	Напруга бази першого транзистора	Струм колектора першого транзистора	Напруга бази другого транзистора	Струм колектора другого транзистора
0	0						
0	1						
1	0						
1	1						

Визначте, яку функцію реалізує схема на рисунку 5.8. Які функції виконують перемикачі X і Y ? Як вони позначаються на схемах? Поясніть, як працює схема на рисунку 5.8. Як вона зображується на схемах?

Контрольні запитання

1. Яку властивість діода використовують для побудови логічних елементів?
2. У яких режимах може працювати транзистор і як вони формуються?
3. Як працює схема за рисунком 5.5?
4. Як працює схема за рисунком 5.6?
5. Як працює схема за рисунком 5.7?
6. Яку функцію реалізує схема за рисунком 5.8?
7. Де використовуються елементи, побудовані за схемою 5.8?

РОБОТА №6. ДОСЛІДЖЕННЯ ТРИГЕРІВ КОМП'ЮТЕРНИХ СИСТЕМ

Мета роботи: знайомство з призначенням тригерів комп'ютерних систем, їхніми видами і способами побудови. Освоєння технології управління JK-тригера і побудови на його основі тригерів різних типів.

У процесі виконання роботи необхідно:

Ознайомитися з призначенням, принципами побудови та видами тригерів комп'ютерних систем.

Освоїти технологію управління тригерами.

Встановити залежності стану тригера від його вхідних сигналів.

Короткі теоретичні відомості.

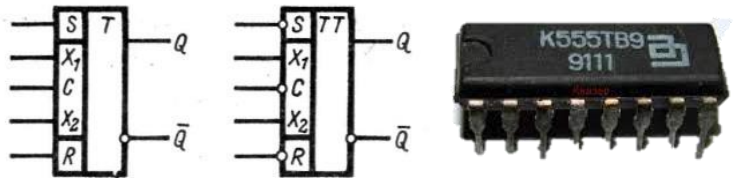
Тригер є елементом, який може знаходитися в одному з двох стійких станів [5]. Одному з цих станів приписується значення 1, а іншому – 0. Це дозволяє йому зберігати один біт інформації. По суті це є найпростішим способом створити пам'ять комп'ютера. Стан тригера розпізнається по його вихідному сигналу. Під впливом вхідного сигналу тригер стрибкоподібно переходить з одного стійкого стану в інший, крім того стрибкоподібно змінюється рівень напруги його вихідного сигналу.

Тригери мають два виходи: прямий Q (називається також «вихід 1») і інверсний («вихід 0»), які знаходяться в протифазі. В одиничному стані тригера на виході Q є високий рівень сигналу, а в нульовому – низький. На виході $\overline{Q}$ навпаки.

Схеми тригерів можна розділити на кілька типів: з установочними входами – RS-тригер, з рахунковим входом – Т-тригер, а також D-тригер, JK-тригер та інше.

Якщо хоча б з одного входу інформація в тригер заноситься примусово під впливом синхронізуючого сигналу, то тригер називається синхронізованим (синхронним). Якщо занесення інформації в тригер з будьякого входу проводиться без синхронізуючого сигналу, то тригер називається несинхронізованим (асинхронним).

Загальна форма умовного позначення тригерів показана на рисунку 6.1. В основному полі ставиться символ Т або символ ТТ для позначення відповідно однотактного або двотактного тригера. Додаткове поле може бути розділене на дві частини: асинхронну і синхронну. У першій проставляються символи R і S входів несинхронізованої установки тригера в «1» і «0», у другій – на місцях X1 і X2 символи відповідно до типу тригера.



а б в

Рисунок 6.1 – Умовні позначення інтегральних тригерів:
а) – синхронізований однотактний тригер із додатковими входами S і R несинхронізованій установки; б) – синхронізований двотактний тригер із додатковими входами S і R несинхронізованій установки; в) – інтегральне виконання тригера К555ТВ9

Користуються такими позначеннями для входів:

S – вхід установки тригера в 1 (від англійської Set);

R – вхід установки тригера в 0 (від англійської Reset);

T – вхід тригера зі рахунковим входом (від англійської Toggle);

D – вхід D-тригера (від англійської Delay);

J – вхід для синхронізованої установки стану «1» в JK-тригері (Від англійської Jump);

K – вхід для синхронізованої установки стану «0» в JK-тригері (від англійської Kill);

C – вхід синхронізації (від англійської Clock).

Якщо вхід відзначений кружком, то це означає, що діюче значення вхідного сигналу – 0.

Наприклад, відсутність кружка на вході C на рисунку 4.1, а вказує на те, що вхідна інформація заноситься в тригер при одиничному значенні синхронізуючого сигналу (діюче значення синхронізуючого сигналу дорівнює 1); кружок на вході C (рисунком 6.1, б) означає, що прийом інформації відбувається при нульовому значенні синхронізуючого сигналу (діюче значення синхронізуючого сигналу дорівнює 0).

Стан тригера визначається сигналом Q на прямому виході тригера (або сигналом на його інверсному виході) $\bar{Q}$.

Закони функціонування тригерів задаються таблицями переходів із компактним записом, за якого в стовпці станів може бути зазначено, що новий стан збігається з попереднім або є його запереченням.

Найпростішим видом тригерів є асинхронний RS-тригер. Схема асинхронного (несинхронізованого) RS-тригера на елементах АБО-НІ подана на рисунку 6.2.

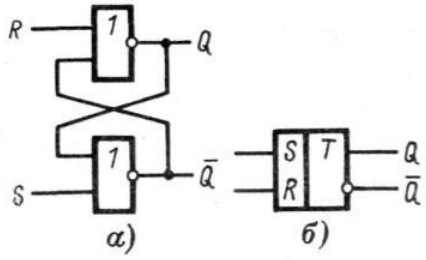


Рисунок 6.2 – Асинхронний RS-тригер на елементах АБО-НІ:
а) – функціональна схема; б) – умовне позначення

Тригер утворений двома комбінаційними схемами АБО-НІ, з’єднаних так, що виникають додатні зворотні зв’язки, завдяки яким у стійкому стані вихідний транзистор однієї схеми АБО-НІ закритий, а в іншої відкритий. Таблиця 4.1, яка називається таблицею переходів, визначає закон функціонування цього тригера.

Таблиця 6.1 – Таблиця переходів асинхронного RS-тригера на елементах АБО-НІ

R	S	Q	Функція
0	0	Q	Зберігання
0	1	1	Установка 1
1	0	0	Установка 0
1	1	-	Заборонено

Якщо R = 1 і S = 0, тригер встановлюється в нульовий стан Q = 0; якщо R = 0 і S = 1, він встановлюється в одиничний стан Q = 1; якщо R = S = 0, тригер зберігає стан, у якому він перебував до моменту надходження на його входи нульових сигналів. Якщо R = S = 1, на прямому та інверсному виходах встановлюється нульовий сигнал. Тригерне кільце перетворюється в два незалежних інвертора, і при переході до зберігання (R = S = 0) тригер може встановлюватися в будь-який стан. Тому така комбінація вхідних сигналів заборонена.

На рисунку 4.3 приведена схема синхронізованого однотактного RS-тригера на елементах І-НІ.

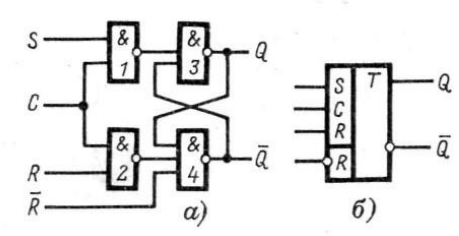


Рисунок 6.3 – синхронізований однотактний RS-тригер на елементах І-НІ: а – функціональна схема; б – умовне позначення

Multisim online немає RS-тригеру як єдиного елементу. Однак його можна з створити самостійно за схемою, що наведена на рисунку

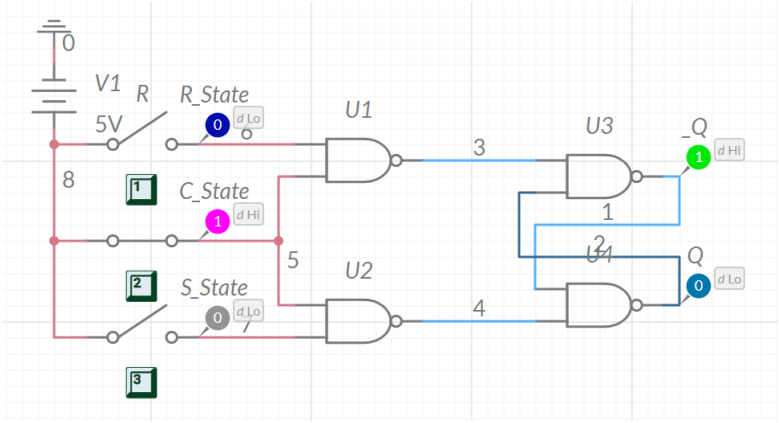


Рисунок 6.4 – Схема синхронізованого RS-тригеру на елементах І-НІ.
Завантажити схему можна за посиланням: http://surl.li/ljprj

Тут елементи 1 і 2 утворюють схему вхідної логіки асинхронного RS-тригера, побудованого на елементах 3 і 4. Такі RS-тригери мають два інформаційних входи R і S і вхід синхронізації С. Крім того, тригер може мати несинхронізовані входи R і S. У цьому випадку функціонування тригера здійснюється або під впливом несинхронізованих входів при С = 0, або під впливом синхронізованих входів. В останньому випадку на несинхронізованих входах мають бути присутніми сигнали, які не впливають на стан схеми.

Стійка робота однотактних RS-тригерів у схемі з передачею інформації між тригерами можлива тільки в разі, якщо занесення в тригер інформації здійснюється після

завершення передачі інформації про його колишній стан в інший тригер. Це просто забезпечується за використання двох серій синхросигналів, які знаходяться в протифазі.

Такий принцип обміну інформацією реалізований в двотактних RS-тригерах. Найпростіша схема двотактного RS-тригера показана на рисунку 6.5.

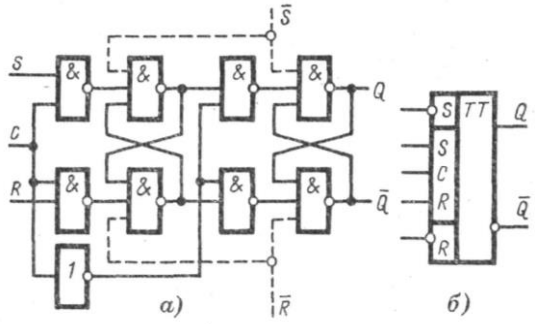


Рисунок 6.5 – Двотактний RS-тригер на елементах І-НІ:
а) – функціональна схема; б) – умовне позначення

Вона складається з двох однотактних RS-тригерів і інвертора в ланцюзі синхронізації. Після надходження на вхід RS-тригера сигналу C = 1 вхідна інформація заноситься в перший однотактний RS-тригер, а другий буде зберігати інформацію, що відноситься до попереднього періоду функціонування. Після закінчення дії сигналу синхронізації, коли C = 0, а, $\bar{C}=1$, перший RS-тригер перейде в режим зберігання, а другий прийме той же стан, що і перший. У результаті до наступного такту на виході двотактного RS-тригера з'явиться сигнал нового стану.

Для установки тригера в стан «0» або «1» без використання $\bar{R},\bar{S}$ синхросигналів у схему вводять додаткові входи *R* і *S* несинхронізованої установки.

Схема RS-тригерів становить основу для побудови інших тригерних схем, таких, як T-, D- і JK-тригери.

Для побудови пристроїв, які здійснюють підрахунок імпульсів – лічильників, використовуються тригери з рахунковим входом (Т-тригер). У найпростішому випадку він може бути побудований з використанням двотактного синхронізованого RS-тригера.

Найпростіша схема несинхронізованого Т-тригера подана на рисунку 6.6, а.

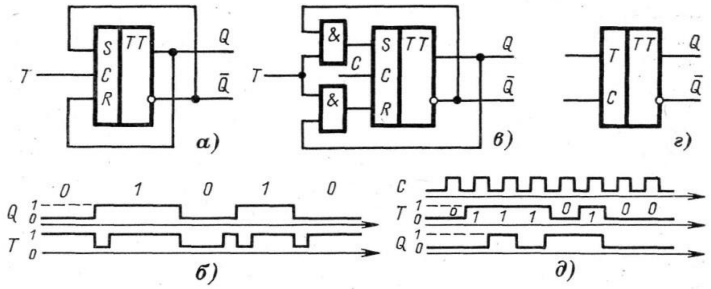


Рисунок 6.6 – Т-тригер:
а) – несинхронізований Т-тригер; б) – часова діаграма роботи несинхронізованого Т-тригера; в) – синхронізований Т-тригер; г) – умовне позначення синхронізованого Т-тригера; д) – часова діаграма роботи синхронізованого Т-тригера

У цій схемі надходження сигналу T = 1 призводить до запису в двоступеневий RS-тригер стану, протилежного раніше збереженого. Разом із тим, оскільки тригер двоступінчастий, на його виході сигнал зміниться лише після завершення дії сигналу T=1, що виключає виникнення генерації у схемі зі зворотним зв'язком. Можна вважати, що в цій схемі одиничний вхідний сигнал подається спадом сигналу T=1, тому що за будь-якої тривалості сигналу T=1 зміна стану T-тригера відбувається тільки один раз – після знятті сигналу T=1 (рисунком 6.6, б).

У разі необхідності представляти потенціалом послідовність одиниць на вході T-тригера використовується синхронізована схема (рисунком 6.6, в, г). Тут одиничний вхідний сигнал представляється високим рівнем сигналу T за C = 1. Тому високим рівнем сигналу T можна уявити послідовність одиниць (рисунком 6.6, д). Запис у тригер відбувається за C = 1, а зміна стану відбувається після закінчення дії сигналу синхронізації C = 1. За T = 1 стан тригера змінюється на протилежний, а за T = 0 стан тригера не змінюється. Часова діаграма, яка пояснює роботу синхронізованого T-тригера, показана на рисунку 6.6, д.

Одним із найбільш широко вживаних тригерів є D-тригер, який реалізує функцію тимчасової затримки. D-тригер має тільки режими установки «1» і «0». У зв'язку з цим несинхронізований D-тригер (рисунком 6.7, а) не застосовується, тому що його вихід буде просто повторювати зовнішній вхід.

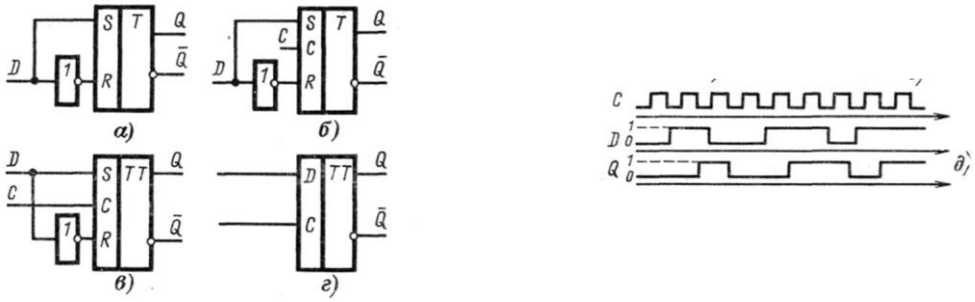


Рисунок 6.7 – D-тригер:
а) – несинхронізований D-тригер; б) – синхронізований однотактний D-тригер; в) – двотактний D-тригер; г) – умовне позначення двотактного D-тригера; д) – часова діаграма роботи двотактного D-тригера

Синхронізований однотактний D-тригер (рисунок 6.7б) затримує поширення вхідного сигналу на час паузи між синхросигналами (затримка на півперіоду). Для затримки на період (на один такт) використовується двотактний D-тригер. Варіант побудови двотактного D-тригера показаний на рисунку 6.7, в. Під дією синхросигналу інформація, яка надходить на вихід D, приймається в RS-тригер, але на виході Q з'являється з затримкою на такт:

$Q(t + 1) = D(t)$.

Універсальним типом тригерів є JK-тригер. Він відрізняється від RS-тригера насамперед тим, що в ньому усунена невизначеність, яка виникає в RS-тригері за комбінації вхідних сигналів R = S = 1. Універсальність JK-тригера полягає в тому, що він може виконувати функції RS-, T- і D-тригерів.

На рисунку 6.8 наведена схема однотактного JK-тригера.

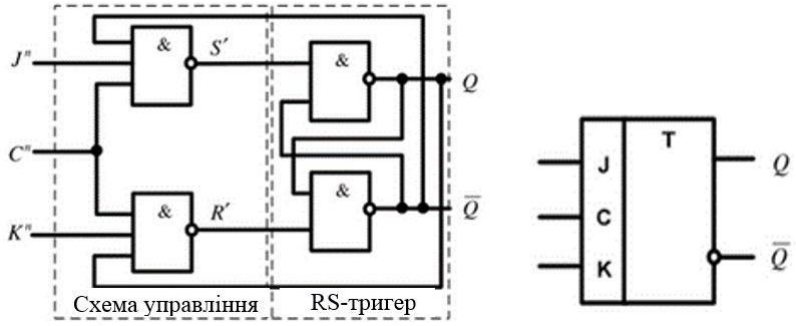


Рисунок 6.8 – Однотактний JK-тригер

Входи J і K відповідають входам установки, відповідно, в «1» і в «0» тригера. JK-тригер на рисунку 6.7 отримано з синхронного RS-тригера шляхом введення додаткових зворотних зв'язків з виходів тригера на входи, які дозволяють усунути невизначеність у таблиці станів. На рисунку 6.7, б зображено умовне позначення цього JK-тригера. На відміну від RS-тригера в JK-тригері сигнали 1 можуть одночасно прийти на входи J і K. Разом із цим стан тригера змінюється на протилежний, тобто за J = K = 1 схема веде себе, як тригер з рахунковим входом. Функціонування JK-тригера може бути описано таблицею 6.2, до того ж такти t тут, як і раніше, задаються сигналом C = 1.

Таблиця 4.2 – Таблиця переходів JK-тригера

t		t+1	
J	K	Q	Функція
0	0	Q(t)	Зберігання
0	1	0	Установка 0
1	0	1	Установка 1
1	1	Q(t)	Інверсія

Поширеним типом тригера в системах інтегральних логічних елементів є двотактний JK-тригер, умовне позначення якого показано на рисунку 6.8, а.

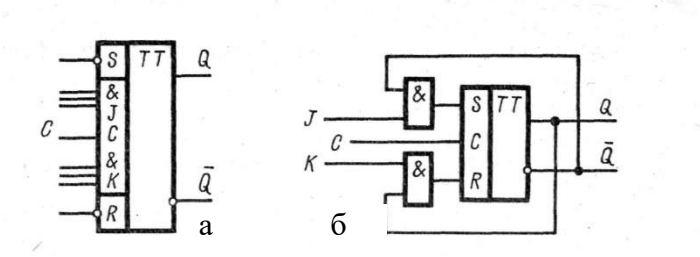


Рисунок 6.9 – JK-тригер:
а) – умовне позначення двохтактного JK-тригера,
б) — схема JK-тригера

Тут J = $J_1J_2J_3$ і K = $K_1K_2K_3$. У розглянутого тригера є входи несинхронізованої установки R і S, за допомогою яких за C = 0 тригер може бути встановлений у стан «1» шляхом подачі R = 1 і S = 0 або в стан 0 шляхом подачі R = 0 і S = 1. Після подачі сигналів

R = S = 1, які не змінюють стан схеми, робота тригера здійснюється під впливом синхронізованих входів.

JK-тригер зручний тим, що за різних варіантах підключення його входів можна отримати схеми, що функціонують як RS-, D- і T-тригери
(рис. 4.9).

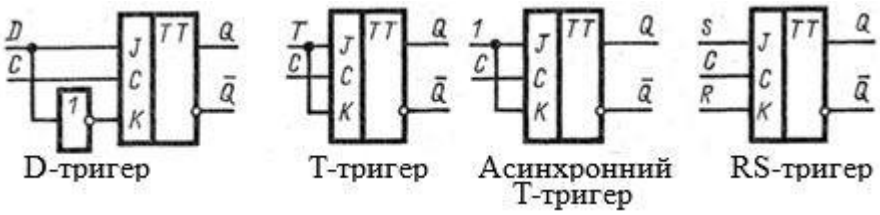


Рисунок 6.10 – Способи використання JK-тригера

Для JK-тригера, який працює в режимі RS-тригера, заборонена комбінація вхідних сигналів R = S = 1.

Порядок виконання роботи

1. Створіть у додатку Word документ для формування звіту, занесіть у нього назву лабораторної роботи та реквізити виконавця.

2. Зберіть схему за рисунком 6.11 і призначте управління перемикачам J, K, C.

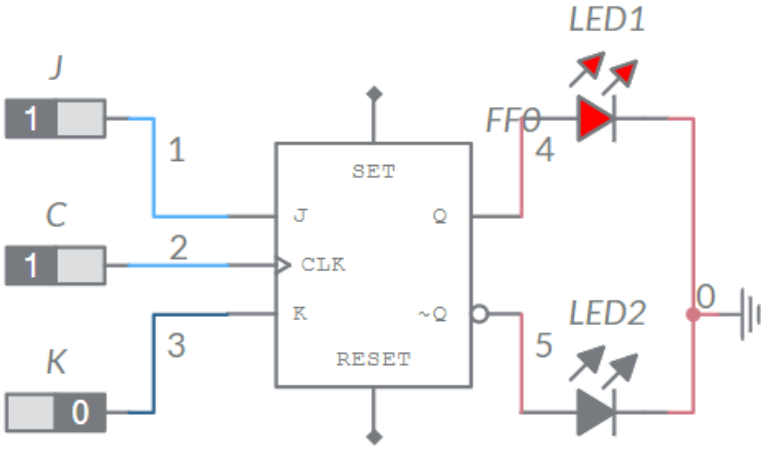


Рисунок 6.11 – Схема підімкнення JK-тригера для першої серії експериментів
(Посилання на заготовку http://surl.li/lkeal)

3. Подаючи по черзі на входи «?» і «/» сигнали 1, встановіть, що це за входи і переназвіть їх на схемі. Скопіюйте схему інструментом «Ножниці» у звіт. Скопіюйте в звіт незаповнені таблиці 6.1, 6.2, 6.3.

4. Примусово встановлюючи тригер в одиничний або нульовий стан і подаючи на входи J і K керуючі сигнали, а на вхід C синхросигнали згідно з таблицями 6.3, 6.4, заповніть колонки 5, 6. Як функцію виберіть одну з наступних: «Зберігання», «Установка в 0», «Установка в 1».

Таблиця 6.3 – Результати виконання п. 4. за одиничного значення тригера в момент часу t

t				t+1	Функція
J	K	C	Q	Q	
0	1	0	1		
0	1	1	1		
0	1	0	1		
Тут ваші висновки					
1	0	0	1		
1	0	1	1		
1	0	0	1		
Тут ваші висновки					
1	1	0	1		
1	1	1	1		
1	1	0	1		
Тут ваші висновки					

Таблиця 6.4 – Результати виконання п. 4 за нульового значення тригера в момент часу t

t				t+1	Функція
J	K	C	Q	Q	
0	1	0	0		
0	1	1	0		
0	1	0	0		
Тут ваші висновки					
1	0	0	0		
1	0	1	0		
1	0	0	0		
Тут ваші висновки					
1	1	0	0		
1	1	1	0		
1	1	0	0		
Тут ваші висновки					

5. Сформулюйте таблицю 6.5, об’єднуючи дані з таблиці 6.1 і таблиці 6.2. Інверсію Q позначайте як invQ.

Таблиця 6.5 – Об’єднана таблиця експериментів за п. 4.

t				t+1
J	K	C	Q	Q
0	0	0		
0	0	1		

0	0	0		
0	1	0		
0	1	1		
0	1	0		
1	0	0		
1	0	1		
1	0	0		
1	1	0		
1	1	1		
1	1	0		

Сформулюйте висновки:

1. Призначення входу K, J, C.
2. Як функціонує тригер під впливом сигналу C?
3. Якщо на входи K і J подати 0 і 0, то стан тригера ...
4. Якщо на входи K і J подати 1 і 1, а на C 1, то стан тригера ...

6. Зберіть схему за рисунком 6.12, використовуючи заготовку (http://surl.li/lkeal), і призначте управління кожному перемикачу. Позначте асинхронні входи. Скопіюйте схему в звіт. Скопіюйте в звіт незаповнені таблиці 6.6, 6.7.

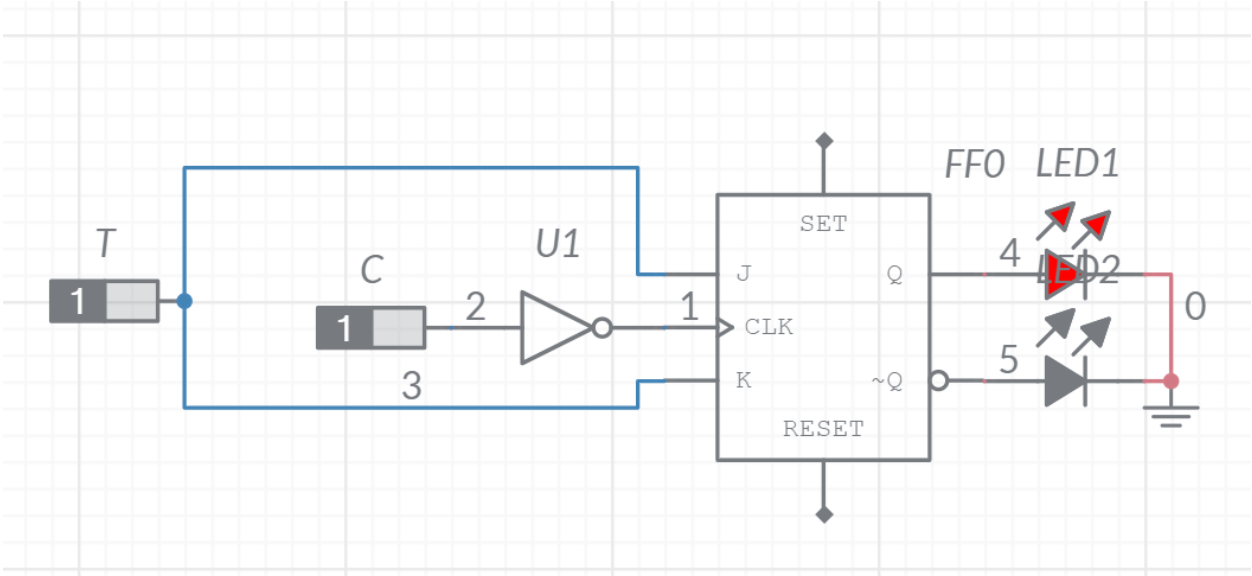


Рисунок 6.12 – Схема підімкнення JK-тригера для другої серії експериментів. Посилання на заготовку http://surl.li/lkeal

7. Примусово встановлюючи тригер в одиничний або нульовий стан і подаючи на вхід C 0 або 1, а на вхід T інформаційні сигнали згідно з таблицею 6.6, заповніть колонку 4. Інверсію Q позначайте як invQ

Таблиця 6.6 – Результати виконання п. Ж

t			t+1
C	T	Q	Q
0	0	1	
0	1	1	
0	0	1	
1	0	1	
1	1	1	
1	0	1	
0	0	0	
0	1	0	
0	0	0	
t			t+1
U	T	Q	Q
1	0	0	
1	1	0	
1	0	0	

З. Узагальніть результати експерименту у таблицю 4.5.

Таблиця 6.7 – Таблиця переходів Т-тригера

t			t+1
U	T	Q	Q
0	0		
0	1		
0	0		
1	0		
1	1		
1	0		

И. Сформулюйте: у який тип тригера перетворюється JK-тригер після об’єднання входів J і K і поданні на них логічної одиниці. Як функціонує тригер у цьому режимі і як він називається?

К. Сформуйте звіт по заняттю, роздрукуйте його і захистить роботу викладачеві.

Контрольні запитання та завдання

1. Що таке тригер?
2. Для чого використовуються тригери?
3. Назвіть види тригерів.
4. Що таке асинхронний тригер?
5. Що таке синхронний тригер?

6. Що таке однотактний тригер?
7. Що таке двотактний тригер?
8. Поясніть за схемою роботу однотактного асинхронного RS-тригера.
9. Поясніть за схемою роботу двотактного синхронного RS-тригера.
10. Доповніть: тригери бувають RS виду, ... Чим вони відрізняються один від одного ?.

РОБОТА №7. ПОБУДОВА І ДОСЛІДЖЕННЯ ЛІЧИЛЬНИКІВ

Мета роботи. Знайомство з призначенням лічильників комп'ютерних систем, їхніми видами і способами побудови. Набуття навичок розробки лічильників різних типів.

У процесі виконання роботи необхідно. Ознайомитися з призначенням, принципами побудови та видами лічильників комп'ютерних систем. Освоїти технологію розробки підсумкових і віднімальних лічильників. Дослідити динаміку функціонування лічильників.

Короткі теоретичні відомості

Лічильником називається вузол комп'ютера, призначений для підрахунку кількості вхідних імпульсів. Лічильники використовуються в комп'ютері для формування послідовностей адрес команд, для підрахунку кількості циклів виконання операцій, для обліку часу і тому подібного. Фактично лічильник становить сукупність з'єднаних у певний спосіб Т-тригерів. Основний параметр лічильника – модуль рахунку (або коефіцієнт перерахунку). Це кількість імпульсів, що надійшли на вхід, яка повертає лічильник у початковий стан.

Якщо M – модуль рахунку, то n – число тригерів для побудови двійкового лічильника і розраховується за такою формулою:

$$n = \text{ОКРУГЛВВЕР}\ [\log_2 (M)] \qquad (7.1)$$

Лічильники позначають через СТ (від англ. Counter). На рисунку 7.1 показано зображення 14-ти розрядного лічильника на схемах (так зване умовне графічне зображення – УГЗ).

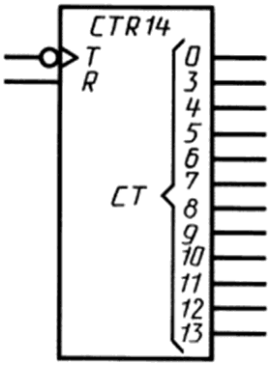


Рисунок 7.1 – Умовне графічне зображення
14-ти розрядного двійкового лічильника

Лічильники імпульсів *класифікують*:

- за системою рахунку:
  - двійкові;
  - двійково-десяткові;
  - з довільним постійним модулем рахунку;
  - зі змінним модулем рахунку;
- за напрямком рахунку:
  - підсумкові;

- віднімальні;
- реверсивні;

- за способом формування внутрішніх зв’язків:
- із послідовним перенесенням;
- із паралельним перенесенням;
- із комбінованим перенесенням;
- кільцеві.

Підсумковий лічильник із послідовним перенесенням може бути побудований за схемою, наведеною на рисунку 7.2. Тут С – вхід, за яким подаються імпульси логічного нуля (у стані простою по лінії С подається логічна одиниця).

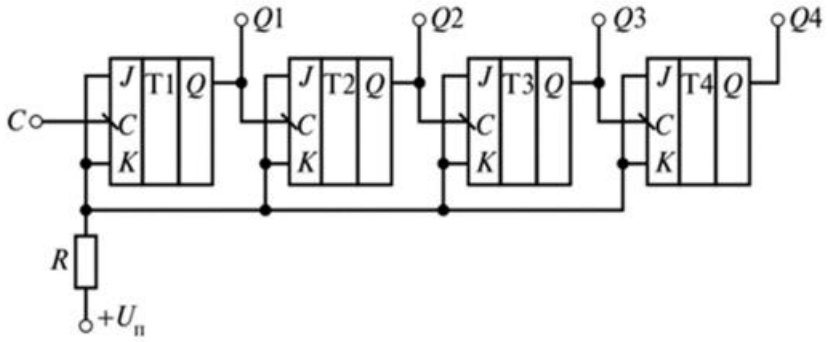


Рисунок 7.2 – Схема підсумкового лічильника на JK-тригерах,
які працюють у режимі Т-тригера

Часова діаграма його роботи наведена на рисунку 7.3. Підрахунок імпульсів починається з нуля. У загальному випадку лічильник працює по циклах. Якщо він заповнений, то імпульс, що знову надійшов, скидає лічильник у нульовий стан.

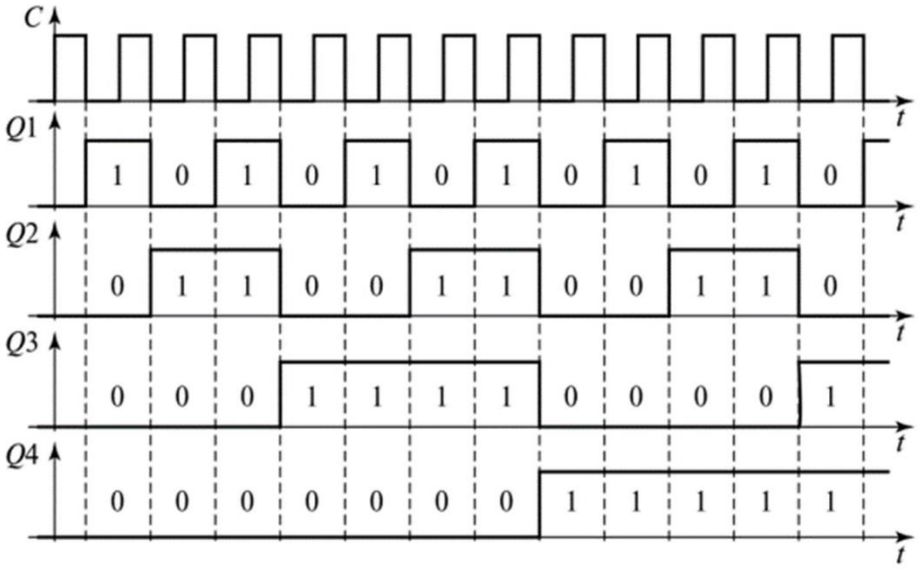


Рисунок 7.3 – Часова діаграма роботи підсумкового лічильника

Віднімальний лічильник із послідовним перенесенням може бути побудований за схемою, наведеною на рисунку 7.4.

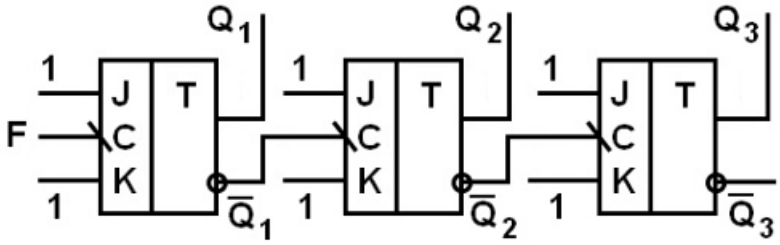


Рисунок 7.4 – Схема віднімального лічильника

Як і в попередньому випадку, лічильник за цією схемою, працює поциклах.

Порядок виконання роботи

А. Створіть документ для формування звіту, занесіть у нього назву лабораторної роботи, реквізити виконавця.

Б. На основі рисунка 7.2 розробіть електричну схему підсумкового лічильника з модулем рахунку *16*, використовуючи елементи з бібліотеки програми Multisim (http://surl.li/lkeal). Вона повинна відповідати таким вимогам:

1. лічильник повинен мати вхід примусової установки всіх тригерів у нуль;
2. рахунок імпульсів повинен йти в природному порядку, тобто послідовна зміна станів після надходження імпульсів така: 0000, 0001,00010, 0011 і т. д.;
3. до одиничного виходу кожного тригера під’єднайте індикатор;
4. розряд кожного лічильника має бути позначений;
5. подача імпульсів здійснюється увімкненням і вимиканням ключа, під’єднаного до джерела напруги;
6. за зразок візьміть схему на рисунку 7.5.

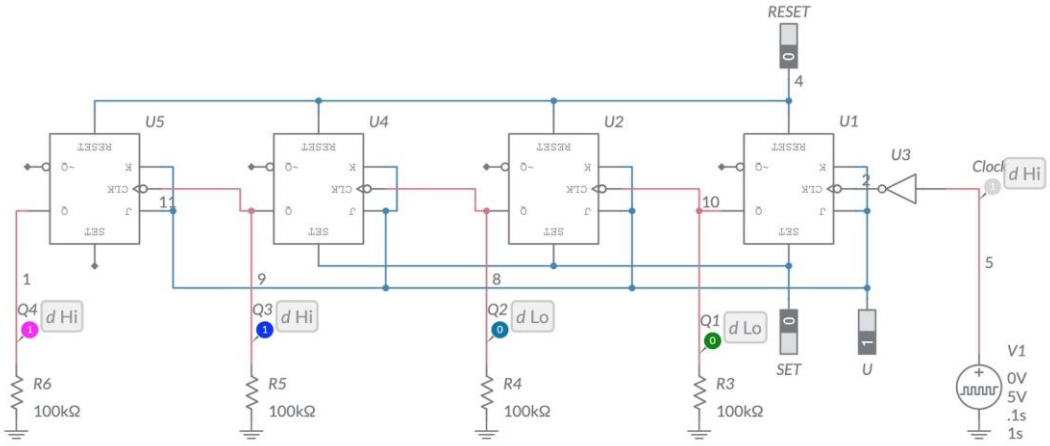
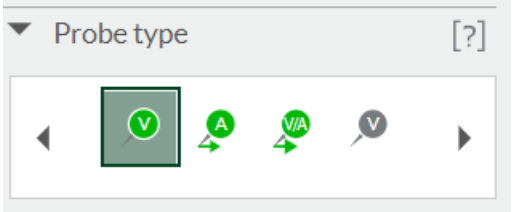


Рисунок 7.5 – Зразок схеми підсумкового лічильника в програмі Multisim

В. Зберіть схему. Як індикатори станів окремих тригерів використовуйте елемент Probe (Digital Probe).



Імпульси на вхід першого тригера подавати вручну. Перевірте працездатність схеми. Працездатну схему скопіюйте в звіт.

Г. Під'єднайте до виходів Q1, Q2, .. тригерів Probe для виводу результатів у вкладку Grapher (аналог багатоканального осцилографа) Grapher. Приклад підімкнення наведено на рисунку 7.5. Скопіюйте схему у звіт.

Обнуліть лічильник. Подавши вручну п'ять імпульсів по входу з частотою 1 Hz, скопіюйте в звіт отримані епюри по входу лічильника і по виходу кожного тригера (приклад на рисунку 7.6). Поясніть роботу лічильника в цьому режимі.

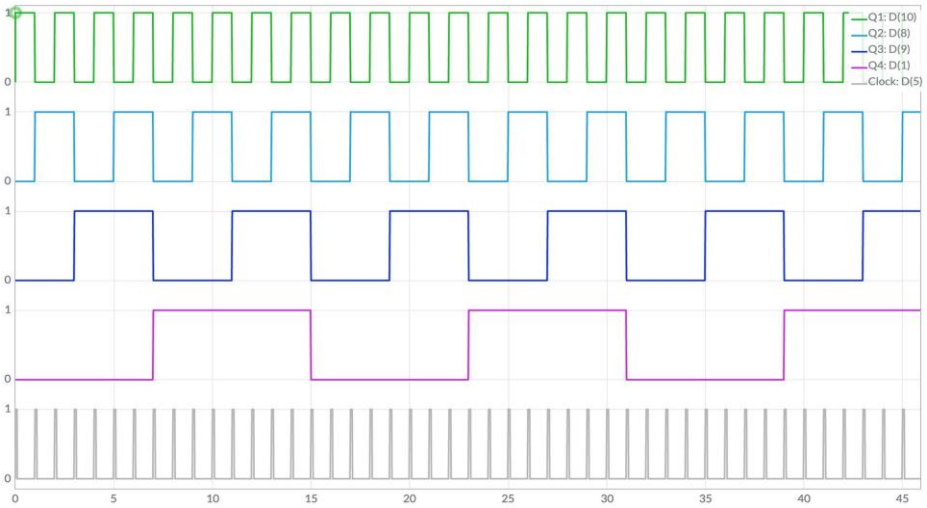


Рисунок 7.6 - Приклад епюр по входу лічильника і по виходах кожного тригера

Ґ. На основі рисунка 7.3 розробіть електричну схему віднімального лічильника з модулем рахунку 16 (приклад схеми наведено на рисунку 7.7), використовуючи елементи з бібліотеки.

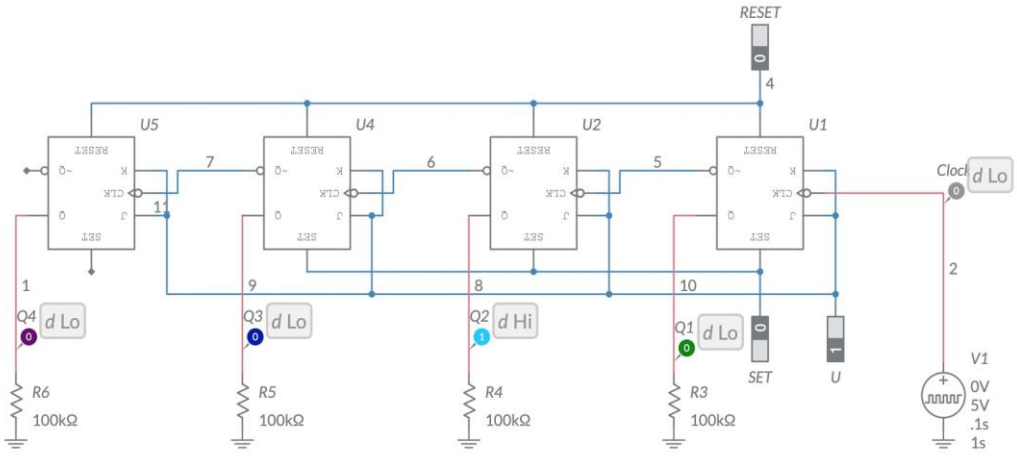


Рисунок 7.7 – Зразок схеми віднімального лічильника в програмі Multisim

Схема повинна відповідати таким вимогам:

1. лічильник повинен мати вхід примусової установки всіх тригерів в одиницю;
2. рахунок імпульсів повинен йти в зворотному порядку, тобто послідовна зміна станів після надходження імпульсів така: 1111, 1110, 1101, 1100 і т. д.

Д. Зберіть схему. Як індикатори станів окремих тригерів використовуйте елемент Probe (аналогічно минулій схемі). Імпульси на вхід першого тригера подавати вручну. Перевірте працездатність схеми. Працездатну схему скопіюйте в звіт.

Е. Під'єднайте до виходів Q1, Q2, .. тригерів Probe для виводу результатів у вкладку Grapher (аналог багатоканального осцилографа) Grapher. Приклад підімкнення наведено на рисунку 7.5. Скопіюйте схему у звіт.

Є. Змініть частоту вхідного сигналу і з'ясуйте, як вона впливає на дослідження. Приклад результату роботи від'ємного лічильника наведено на рисунку 7.8.

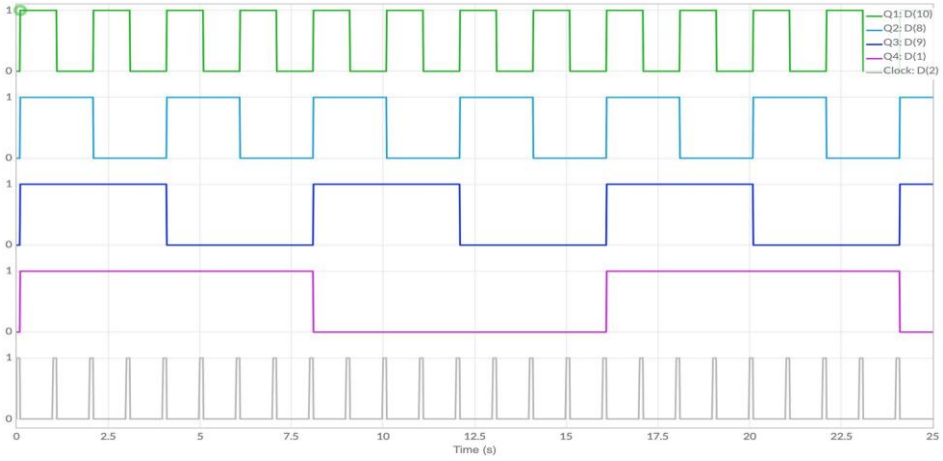


Рисунок 7.8 - Приклад результату роботи від'ємного лічильника

Ж. Надайте в звіті відповіді на контрольні питання нижче.

Контрольні запитання і завдання

1. Що таке лічильник?
2. Для чого використовується лічильник у комп'ютері?
3. Що таке модуль рахунку і як його розрахувати?
4. Тригери яких типів і в яких режимах можна використовувати для побудови лічильників?
5. Які бувають види лічильників і в чому їхні особливості?
6. Для чого можна використовувати Grapher і як його налаштувати?
7. Поясніть роботу лічильника за часовою діаграмою.
8. Як сформувати цілісний вузол комп'ютера і де його можна використовувати?

РОБОТА №8. ДОСЛІДЖЕННЯ АРИФМЕТИКО-ЛОГІЧНОГО ПРИСТРОЮ

Мета роботи. Ознайомлення з принципом побудови арифметико- логічного пристрою процесора і технологією його функціонування.

У процесі виконання роботи необхідно. Ознайомитися з принципом побудови і функціонування арифметико-логічного пристрою процесора. Визначати тип операцій, які може виконувати арифметико-логічним пристрій по таблиці істинності. Засвоїти технологію управління арифметико-логічним пристроєм процесора.

Короткі теоретичні відомості

Основним блоком комп'ютера є операційний пристрій, який здійснює виконання команд поточної програми. У найпростішому випадку він складається з арифметико-логічного пристрою (АЛП), пристрою управління, блока регістрів, шин управління і даних. Кожна команда програми містить код операції, який дешифрується в пристрої управління. Результатом цього є комутація схеми АЛП на виконання відповідної команди з даними, які надійшли з регістрів даних. Усе АЛП будується переважно з ідентичних однорозрядних АЛП, кожне з яких виконує всі функції над одним розрядом паралельного слова. На рисунку 8.1 зображена схема такого АЛП спільно з вузлом управління, який виконує обмежений набір операцій.

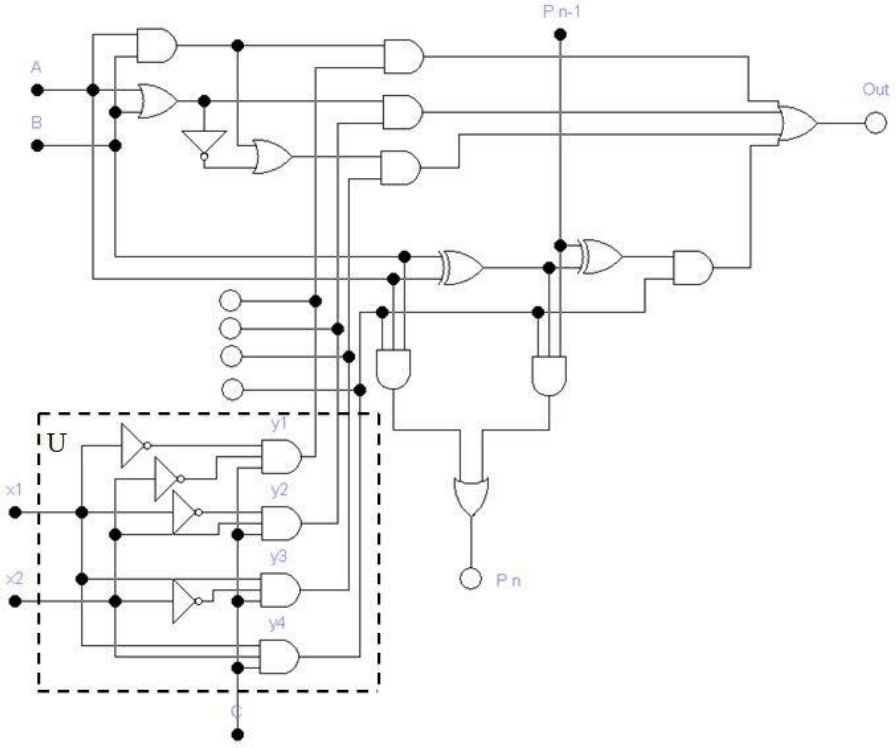


Рисунок 8.1 – Схема однорозрядного АЛП

На входи A і B надходять однойменні розряди двох чисел, що беруть участь в операції. На входи *x*1 і *x*2 надходить код операції, яку необхідно виконати над цими числами. Код операції розпізнається у вузлі U (необхідно буде визначити, що це за вузол), який комутує АЛП на виконання конкретної операції. Запуск операції на виконання здійснюється подачею синхросигналу на вхід C. Результат операції надходить на вихід Out, а в деяких операціях передається через вихід P_n у старший розряд АЛП. Прийом інформації з молодшого розряду АЛП здійснюється через вхід P_{n-1}.

Природно, що сучасні процесори мають у своєму складі АЛП на кілька порядків складніше описаного, але принцип побудови залишається одним і тим же.

Порядок виконання роботи

А. Створіть документ для формування звіту, занесіть у нього назву лабораторної роботи, реквізити виконавця.

Б. Схему АЛП, реалізуйте в програмі Multisim та занесіть у звіт із лабораторної роботи. Для прикладу розглянути схеми на рисунках 8.2-8.4.

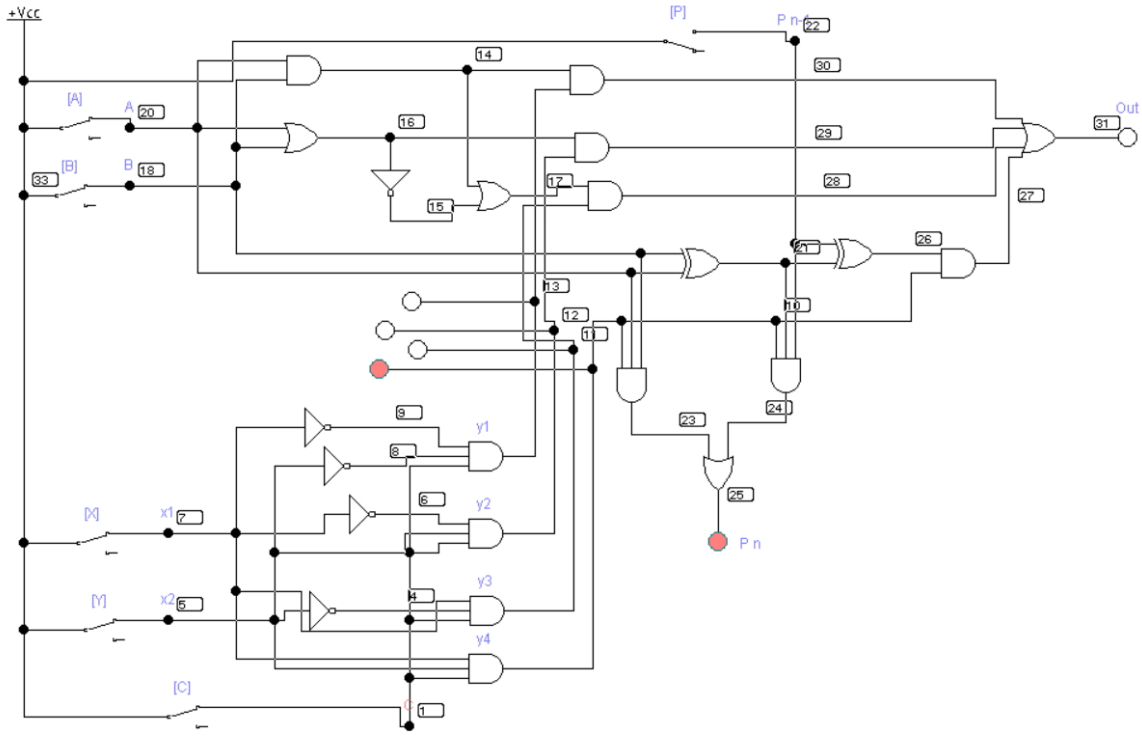


Рисунок 8.2 – Загальна схема однорозрядного АЛП

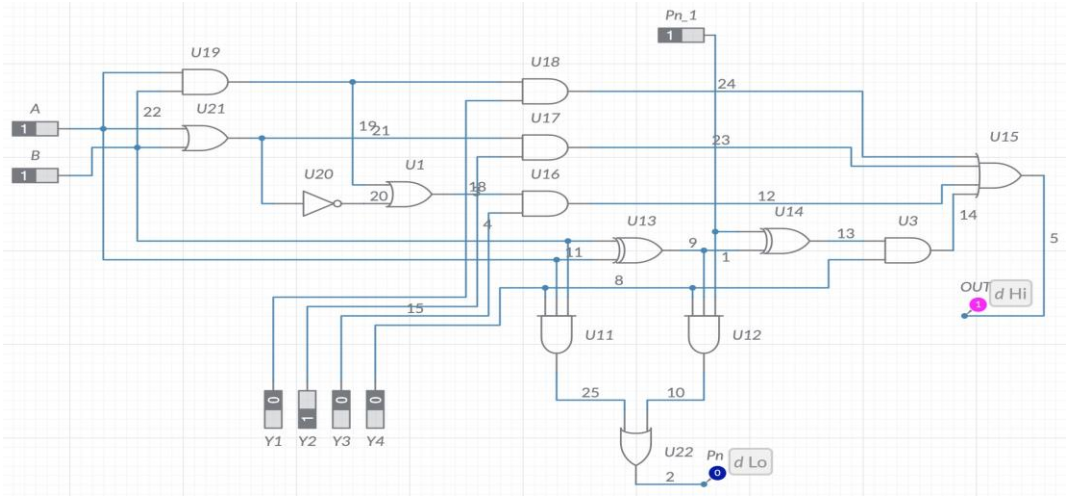


Рисунок 8.3 – Основний блок однорозрядного АЛП

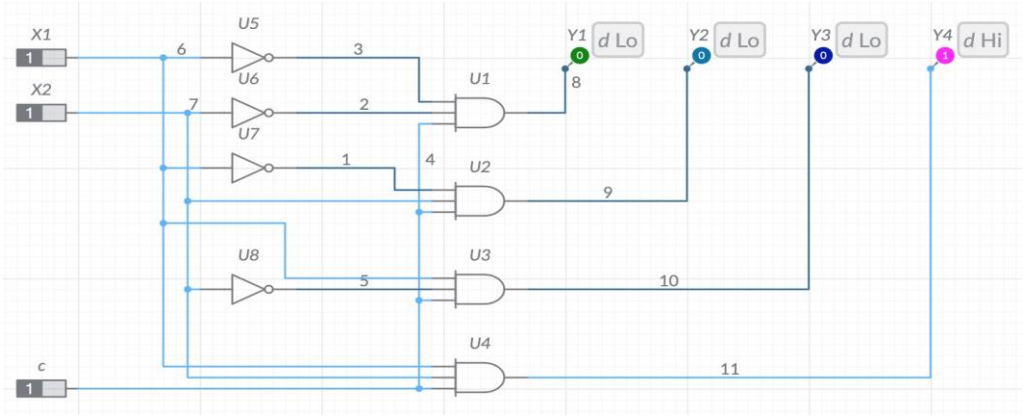


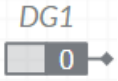
Рисунок 8.4 – Вузол U - розпізнавання коду операції

В. Скопіюйте в звіт незаповнену таблицю 8.1.

Таблиця 8.1 – Таблиця істинності для вузла U

x1	x2	y1	y2	y3	y4
0	0				
0	1				
1	0				
1	1				

Увага. Замикання ключа ініціює подачу логічної одиниці на вхід схеми. Значення одиниці елементу Digital Constant

сигналізує про наявність логічної одиниці на відповідному виході схеми, значення нуля 

сигналізує про наявність логічного нуля на відповідному виході схеми.

Встановіть вхід «С» в одиничний стан. Посилаючи на входи x1 і x2 з джерела поперемінно 0 і 1 (виконати всі можливі комбінації ключів) і фіксуючи значення сигналу на виході схем y1, y2,y3 і y4, заповніть таблицю 8.1, використовуйте елемент

Digital Probe. Значення одиниці елементу сигналізує про наявність логічної одиниці на відповідному виході схеми, значення нуля сигналізує про наявність логічного нуля на відповідному виході схеми.

Визначте, який типовий вузол комп'ютера реалізований блоком U та яку функцію в АЛП він виконує. Відповіді запишіть у звіт.

Г. Встановіть на входах x1 = 0, x2 = 0, C = 1.

Посилаючи на входи A і B з джерел поперемінно 0 і 1 і фіксуючи значення сигналу на виході схеми Out і P_n, аналогічно використовуйте елемент Digital Probe.

Таблиця 8.2 – Таблиця істинності для досліджень за п. Г

A	B	Out	P_n
0	0		
0	1		
1	0		
1	1		

Сформулюйте у звіті, яку операцію виконує АЛП у цьому режимі. Натискаючи на перемикач P_{n-1}, визначте його вплив на роботу.

Ґ. Встановіть на входах x1 = 1, x2 = 0, C = 1.

Посилаючи на входи A і B із джерел поперемінно 0 і 1, фіксуючи значення сигналу на виході схеми Out і P_n в таблиці 8.3.

Таблиця 8.3 – Таблиця істинності для досліджень за п. Ґ

A	B	Out	P_n
0	0		
0	1		
1	0		
1	1		

Сформулюйте у звіті, яку операцію виконує АЛП в цьому режимі. Натискаючи на перемикач P_{n-1}, визначте його вплив на роботу.

Д. Встановіть на входах x1 = 0, x2 = 1, C = 1.

Посилаючи на входи A і B з джерел поперемінно 0 і 1 і фіксуючизначення сигналу на виході схеми Out і P_n, заповніть таблицю 8.4.

Таблиця 8.4 – Таблиця істинності для досліджень за п. Д

A	B	Out	P_n
0	0		
0	1		
1	0		
1	1		

Сформулюйте у звіті, яку операцію виконує АЛП у цьому режимі. Натискаючи на перемикач P_{n-1}, визначте його вплив на роботу.

Е. Встановіть на входах x1 = 1, x2 = 1, C = 1.

Посилаючи на входи A, B і P_{n-1} з джерел поперемінно 0 і 1 і фіксуючизначення сигналу на виходах схеми Out і P_n, заповніть таблицю 8.5.

Таблиця 8.5 – Таблиця істинності для досліджень за п. Е

A	B	P_{n-1}	Out	P_n

0	0	0		
0	1	0		
1	0	0		
1	1	0		
0	0	1		
0	1	1		
1	0	1		
1	1	1		

Сформулюйте у звіті, яку операцію виконує АЛП у цьому режимі.

Є. Надайте в звіті відповіді на контрольні питання нижче.

Контрольні запитання і завдання

1. Що таке АЛП?
2. Який блок процесора управляє АЛП і за допомогою чого?
3. Де зберігається інформація, яка обробляється в АЛП?
4. Як в обчислювальній техніці називається виділена частина АЛП?

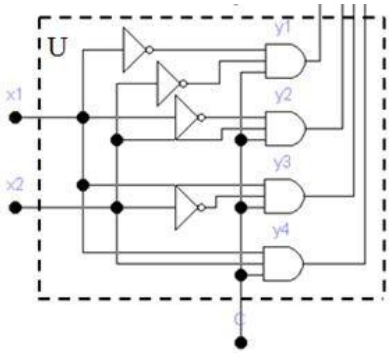


Рисунок 8.5 – Блок U арифметико-логічного пристрою

5. Що в ньому надходить на вхід і що виробляється на виході?
6. На вхід блока U надійшов код 10 (x1 = 1, x2 = 0). Визначте, які сигнали сформуються на виходах y1, y2, y3, y4.
7. Які операції може виконувати досліджений у роботі АЛП?
8. Як здійснюється вибір тієї чи іншої операції, яку виконує АЛП?

РОБОТА №9. НАПИСАННЯ ПРОСТОЇ ПРОГРАМИ НА МОВІ ASSEMBLER

Мета роботи. Ознайомитись з процесом написання та налагодження програми на мові Асемблер.

У процесі виконання роботи необхідно. Вивчити основи оформлення типових програм на асемблері.

Теоретичні відомості

Структура програми на мові асемблер

При вивченні будь-якої мови програмування традиційним є написання програми, що виводить на екран повідомлення “Hello, world!”. Текст такої програми на мові Assembler представлено на лістингу 9.1.

```
.MODEL small
.STACK 4096
.DATA
    Hellostr DB 'Hello, world!', 13, 10, '$'
.CODE
main PROC
    mov ax,@data
    mov ds,ax
    mov dx,offset Hellostr
    mov ah,9h
    int 21h
    mov ah, 04Ch
    int 21h
main ENDP
END main
```

Лістинг 9.1 – Текст програми на мові Assembler

На перший погляд може здатися, що дана програма довша, ніж аналогічна програма, написана на іншій мові. Програми, створені з використанням мови Assembler дійсно довші, оскільки кожна інструкція виконує менше дій, ніж інструкція на мові високого рівня. З іншого боку, мова Assembler дозволяє створювати максимально ефективні програми, які повністю контролюють ресурси обчислювальної системи.

У програмі є місце, де зберігаються дані і де зберігається код, ці місця потрібно розділяти. Директива .DATA вказує на те що далі будуть йти дані а директива .CODE що тепер почнуться команди процесора. Це важливий поділ даних і команд процесора. Коли програма завантажується в пам'ять то операційній системі потрібно знати куди поставити вказівник для виконання команди. Саме директива .CODE і вказує при складанні, де це місцебуде знаходитись. Те ж саме і для .DATA.

Переривання

В програмі ми використали команду INT:

```
mov ah,9h
int 21h
mov ah, 04Ch
int 21h
```

Int - це скорочення слова Interrupt що з англійської на українську переводитися як переривання. Виглядає ця команда так:

Int номер

Тобто у кожного переривання є номер. З'явилося поняття переривання разом із створенням ЕОМ. Тоді стояло завдання про спільну роботу процесора і повільних зовнішніх пристроїв. Хорошим прикладом може служити клавіатура. Коли користувач натисне клавішуне відомо. Це може трапитися в будь-який момент, от коли він натискає клавішу – процесору повідомляється що потрібно обробити цю дію і отримати код клавіші, яка натиснута.

Можна зробити висновок, що переривання породжують зовнішні пристрої. Звичайно, крім отримання інформації від пристрою, цими пристроями потрібно ще й керувати. Пристрої повільні і крім іншого ще потрібно буде дочекатися закінчення виконання операції. Це теж реалізується за допомогою переривань тільки викликаються вони з програми. Отже, переривання бувають двох типів:

- програмні;
- апаратні.

Пристроїв існує багато - клавіатура, монітор, принтер, миша і так далі. Якщо не користуватися перериваннями, то операційна система повинна постійно опитувати пристрої, чи натиснута клавіша, чи потрібно вивести дані на монітор і так далі. Набагато простіше обумовити деякий механізм на який і буде звертатися увага операційної системи і процесора на необхідність проведення деяких дій.

Отже, програма щось виконує. У цей момент натискається клавіша. Програма повинна бути перервана. І як тільки це буде зроблено, управління буде передано спеціальним кодом (процедура обробки переривання), а потім програма буде виконуватися далі.

Те ж саме коли викликаються переривання для виведення символів на екран монітора (int 21h) то переривання самі генеруються. У цей момент може відбуватися зчитування з зовнішнього носія інформації або вивід інших символів на екран. Через те що переривання можуть наступати одночасно, існує пріоритет їх обробки. Є переривання, які будуть виконуватися в будь-якому випадку навіть якщо йде обробка іншого переривання.

Процедура обробки переривання це програма. Питання в тому тільки де вона зберігається. Базова обробка переривання зберігається в BIOS і в самих мікросхемах. Але використовувати їх досить важко. Для того, що б записати файл потрібно завести двигун дисковода, встановити головку в потрібному місці, дати команду перейти в той сектор прочитати таблицю файлів, перевірити чи там немає файлу і так далі. Всі ці завдання полегшує операційна система, яка надає Вам можливість використовувати переривання більш високо рівня. Використовуючи ці переривання, Ви можете одним етапом створити файл, наприклад. Розрізняють переривання за номерами:

21h - переривання DOS

13h - переривання BIOS

Коли викликається переривання (INT) вказується ще його номер (21h) тобто, призначається той пристрій, який буде виконувати цю дію.

Повернемось до коду програми "HelloWord":

```
mov ah, 9h
int 21h
mov ah, 04Ch
int 21h
```

Як видно одне і теж переривання викликаються два рази. У першому випадку виводиться рядок на екрані монітора, у другому закривається програма. Номер переривання один і той самий, так як же вдалося операційній системі розібратися що до чого? Справа в тому, що одне переривання може виконувати багато функцій. Наприклад виведення на екран і завершення програми. Просто викликати INT 21h мало, ще потрібно вказати що Ви конкретно хочете зробити. Ось це вказується в регістрі AX. В регістрі AX вказується що потрібно зробити, тобто яку дію. До речі цей регістр ділитися на дві частини (AX: AH + AL).

```
mov ah, 9h     ; функція виведення рядка
mov ah, 04Ch   ; функція завершення програми
```

А тепер все разом:

```
mov ah, 9h     ; будемо виводити рядок int 21h    ; вивести
mov ah, 04Ch   ; будемо закривати програму int 21h     ; закрити
```

Функцію потрібно вказувати для усіх переривань
(10, 13, 21 і так далі).

Регістри

Регістри це спеціальні комірки пам'яті. Вся їх перевага у тому, що звернення до регістрів здійснюється значно швидше ніж до оперативної пам'яті ПК. Саме з цієї причини регістри використовуються для команд процесора. Звідти процесору зручно і швидко отримувати інформацію. Якщо говорити про ПК з типом процесорів 286 то розмір регістру 16 біт.

Кожен регістр має ім'я і своє призначення. Вони бувають наступні за типами:

- регістри загального призначення: AX, BX, CX, DX, BP, SI, DI, SP
- сегментні регістри: CS, DS, SS, ES, GS, FS
- лічильник команд IP
- регістр прапорів FLAGS

В ПК з 32-розрядними процесорами є такі основні типи регістрів:

- регістри загального призначення: EAX, EBX, ECX, EDX, EBP, ESI, EDI, ESP
- сегментні регістри: CS, DS, SS, ES, GS, FS
- лічильник команд EIP
- регістр прапорів EFLAGS

В ПК з 64-розрядними процесорами є такі основні типи регістрів:

- регістри загального призначення: RAX, RBX, RCX, RDX, RBP, RSI, RDI, RSP
- сегментні регістри: CS, DS, SS, ES, GS, FS
- лічильник команд RIP

- регістр прапорів RFLAGS Кожне ім'я регістра має певне значення:

AX (EAX, RAX) - accumulator акумулятор;BX (EBX, RBX) - base база;

CX (ECX, RCX) - counter лічильник;DX (EDX, RDX) - data дані;

BP (EBP, RBP) - base pointer вказівник бази; SI (ESI, RSI) - source index індекс джерела;

DI (EDI, RDI) - destination index індекс приймача; SP (ESP, RSP) - stack pointer вказівник стека;

CS - code segment сегмент команд; DS - data segment сегмент даних; SS - stack segment сегмент стеку;

ES - extra segment додатковий сегмент; GS - extra segment додатковий сегмент; FS - extra segment додатковий сегмент; IP - instruction pointer лічильник команд.

Регістри AX, BX, CX і DX дозволяють нам звертатися не лише до всього регістра атакож до старшого і молодшого байта:

AX: AH, ALBX: BH, DLDX: DH, DLCX: CH, CL

Структура основних програмних регістрів процесора IA-32 представлена на рисунках 9.2-9.4.

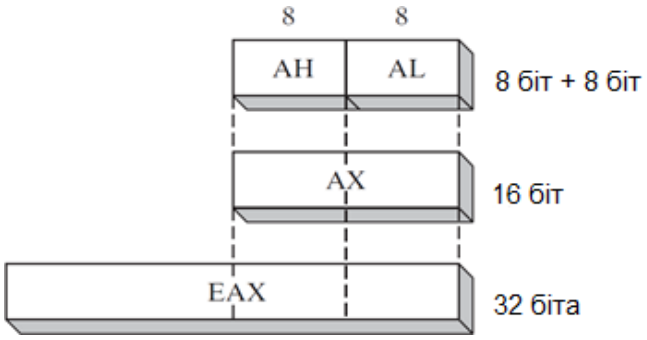


Рисунок 9.2 - Старші і молодші байти регістра

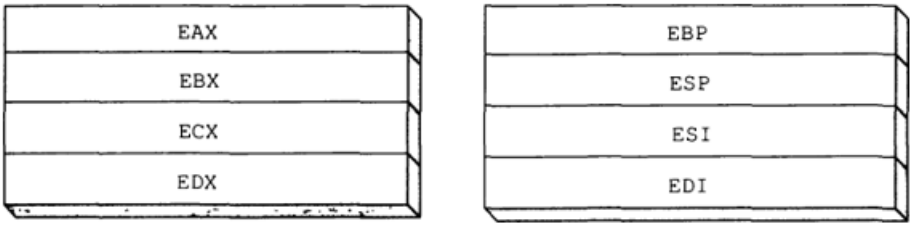


Рисунок 9.3 - 32-розрядні регістри загального призначення

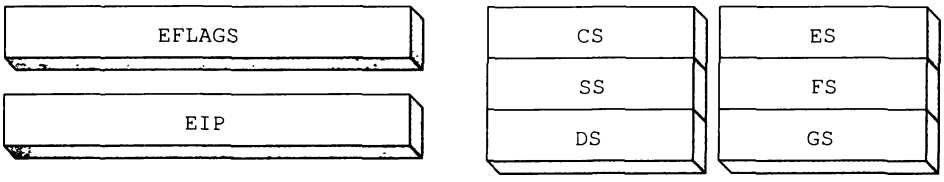


Рисунок 9.4 - 16-розрядні сегментні регістри

В програмі «HelloWord» використовувався регістр AX для задання функції переривання:

```
mov ah, 9h
int 21h
mov ah, 04Ch
int 21h
```

При цьому використовувалась тільки частина регістра, а точніше старший байт:

- H (high) старший
- L (low) молодший

Чому не можна було використати, наприклад, комірки пам'яті? Тому що в асемблері існують правила про те, що і в якому регістрі повинно знаходитися. Правила, ці описані в документації. Наприклад остання функція описана так:

Int 21H Функція 4CHAH = 4CH

AL = код повернення

Повернення немає.

Припиняє процес і передає операційній системі код повернення.

Для більшості операцій використовуються регістри, а для переривань конкретно є специфікації, в яких написано що і в якому регістрі повинно знаходиться.

Команда MOV

В програмі кілька разів зустрічалася команда MOV:

```
mov dx, offset Hellostr
mov ah, 9h
```

Сенс команди MOV полягає в приміщенні з одного місця в інше:

MOV одержувач, передавач

Наприклад:

```
mov dx, offset Hellostr  ; помістити в dx зміщення рядка
mov ah, 9h     ; помістити в ah число 9h
```

Для вказівки сегменту даних використовується регістр DS. Тобто цей регістр повинен вказувати на початок даних в нашій програмі або на сегмент .DATA. В програмі було так:

```
mov ax, @data mov ds, ax
```

По-перше, @data - це ідентифікатор сегменту даних .DATA при компіляції і збірці програми на місце цього слова буде поставлено реальне зміщення сегмента, в якому знаходяться дані. Адже до компонування програми ці інформація не відома. І реально це число стане відоме тільки при компонуванні програми.

Чому інформація поміщається спочатку в регістр AX? Існує правило - не можна безпосередньо змінювати вміст регістрів CS, DS, SS. Тобто ми не можемо написати так:

```
mov ds, 12345
```

Ми можемо змінити цей регістр тільки використовуючи інші регістри. Отже, все разом:

```
mov ax, @data  ; в регістр помістити AX зміщення для сегмента даних
mov ds, ax     ; встановити вміст регістра DS рівним AX тобто тепер
там значення зміщення сегмента даних.
```

Отже, у регістр DS не можливо безпосередньо отримати доступ, а тільки використовуючи інші регістри. Цей регістр вказує на дані.

Отже, регістр DS вказує на сегмент даних, але ж даних може бути багато, наприклад, багато рядків, ось для цього і використовується адресація сегменту - зміщення. Взагалі подібна адресація була створена у зв'язку з необхідністю адресації до більшого діапазону ніж дозволяє число 16 біт. У нашому випадку нам потрібно ще й отримати вказівник на дані. І це було зроблено:

```
.DATASEG
Hellostr DB 'Hello, world!', 13, 10, '$'
.CODESEG
main PROC
......
mov dx, offset Hellostr
```

Символи з кодами 13 і 10 позначають перехід на наступний рядок (символ 13 (0Dh) називається CR - Carriage Return - повернення каретки (курсора) на початок поточного рядка, а символ 10 (0Ah) LF - Line Feed - переведення каретки (курсора) на наступний рядок.

В цьому коді використовувалась директива offset – вона дає змогу обчислювати те як зміщені певні дані відносно початку сегмента даних. У приведеному прикладі це змінна *Hellostr*. При компонуванні програми в це місце буде вставлено конкретне число.

Інформація в регістрах DS:DX буде вказувати на місце розташування змінної *Hellostr* в сегменті даних. Після того як регістр DS ініціалізований, програма отримає першу літеру змінної *Hellostr*. Ось саме з'ясуванням цього числа і займається директива offset.

Отже, у даних є поняття зсув, яке вираховується щодо регістра DS. Для цього є директива offset:

Offset ім'я_змінної

Порядок виконання роботи

I. Встановлення програми emu8086, процес представлено на рисунках 9.5-9.9.

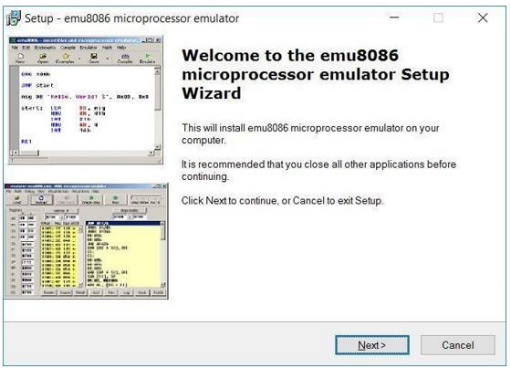


Рисунок 9.5 – Початкове вікно інсталятора програми

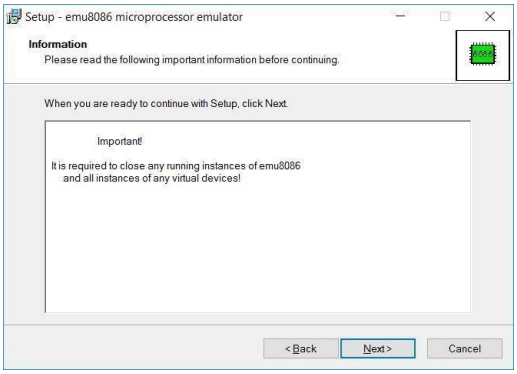


Рисунок 9.6 – Інформаційне вікно інсталятора програми

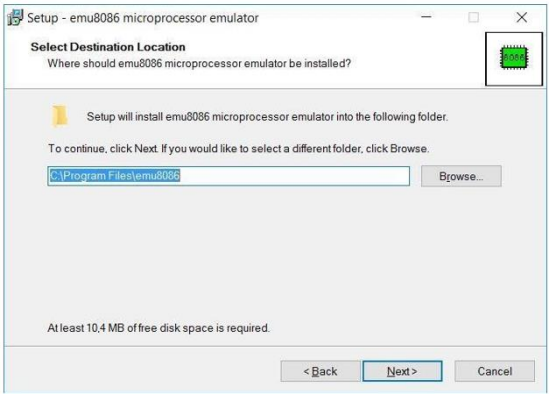


Рисунок 9.7 – Обрання місця інсталяції програми

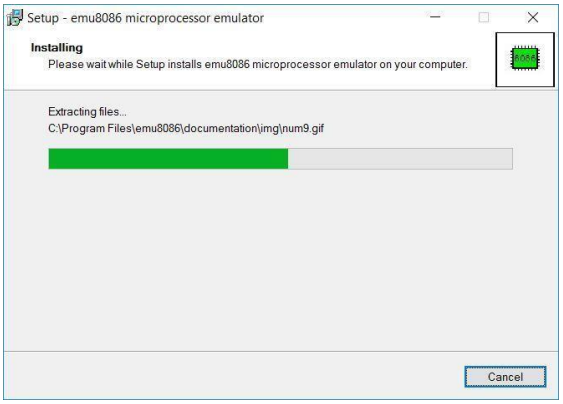


Рисунок 9.8 – Процес інсталяції програми

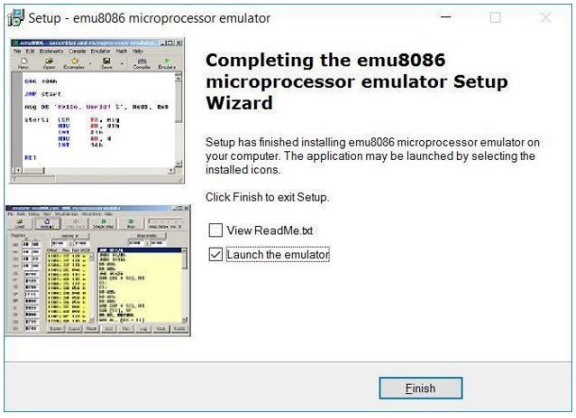


Рисунок 9.9 – Завершення інсталяції програми

II. Процес створення проекту в програмі emu8086 показано на рисунках 9.10-9.12.

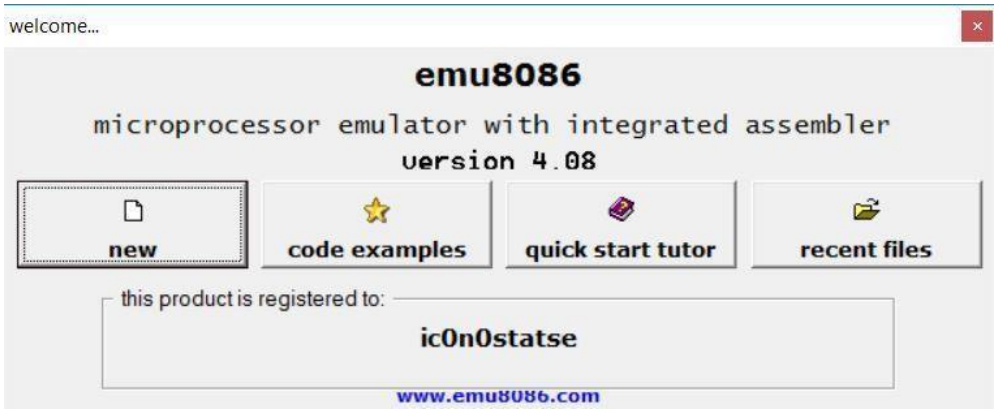


Рисунок 9.10 – Створення нового проекту в програмі emu8086

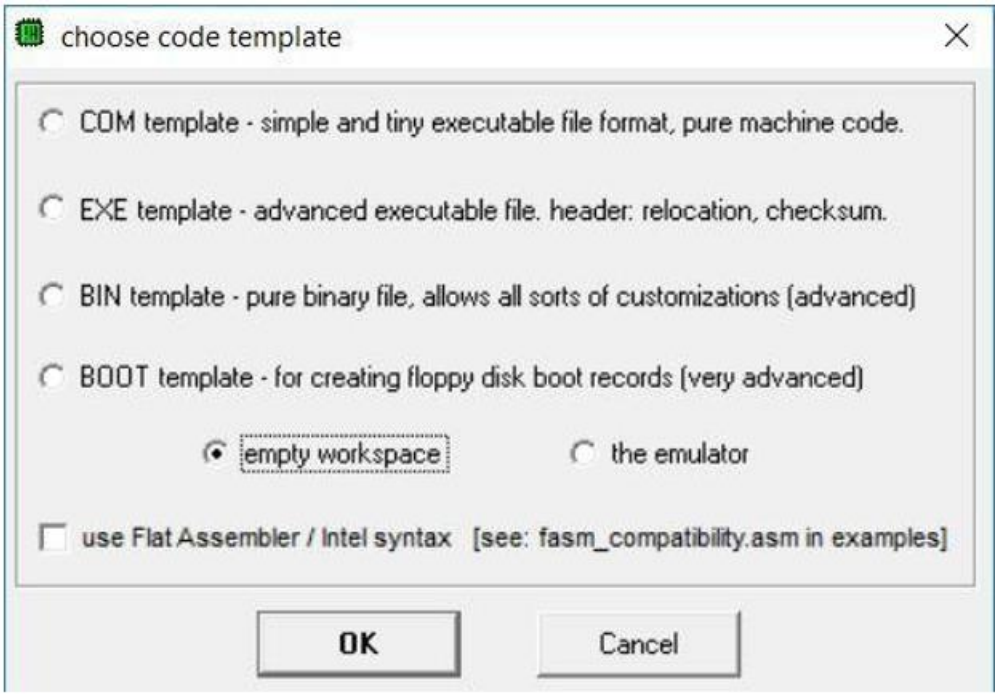


Рисунок 9.11 – Обрання шаблон коду в програмі emu8086

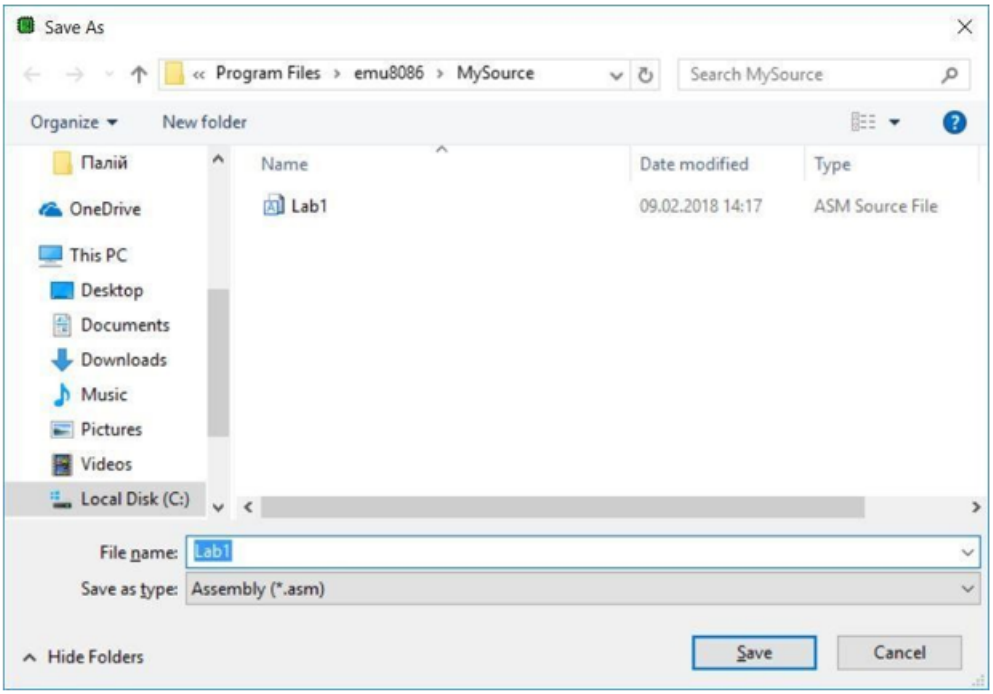


Рисунок 9.12 – Обрання місця збереження проекту
в програмі emu8086

III. Написання коду найпростішої програми мовою Assembler, рисунок 9.13.

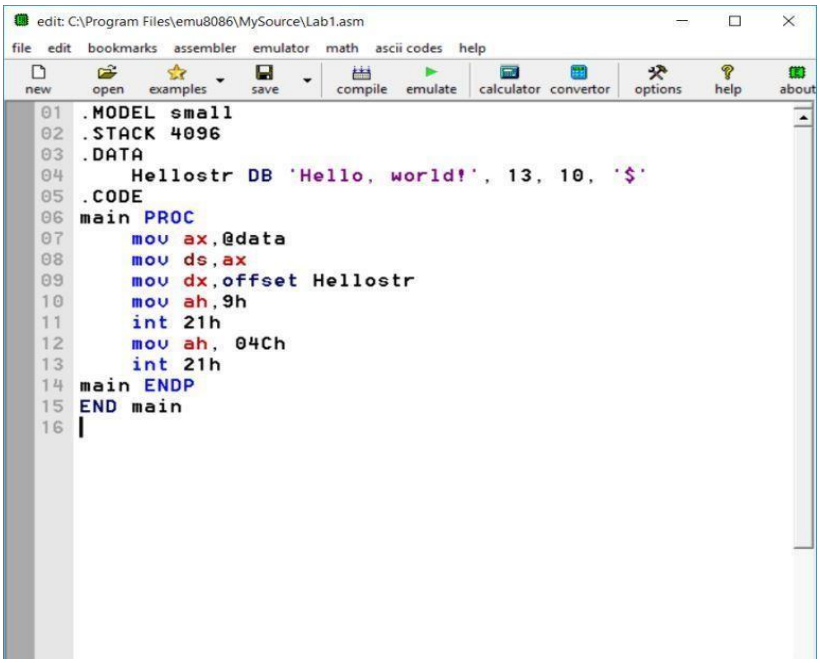


Рисунок 9.13 – Коду найпростішої програми мовою Assembler

IV. Ассемблювання коду, рисунок 9.14.

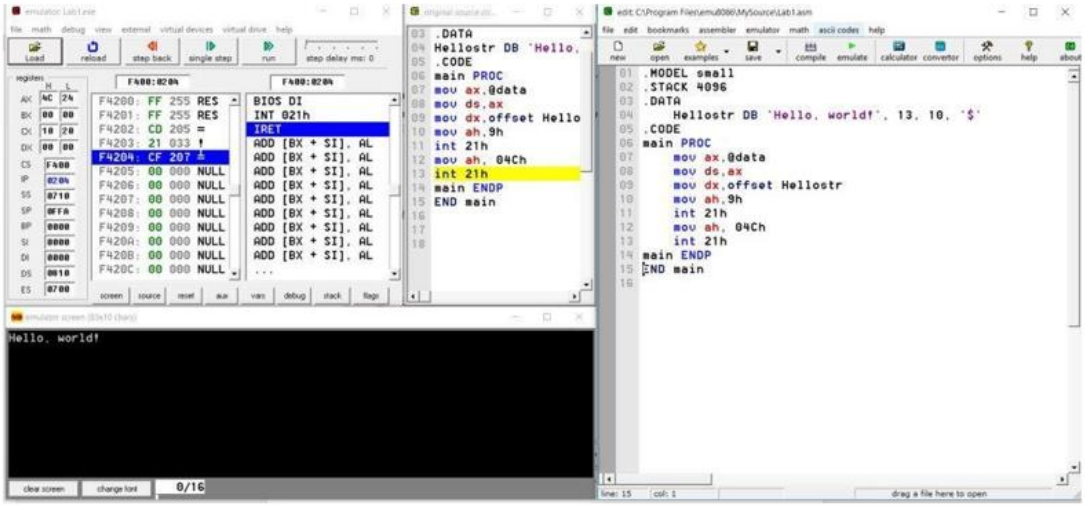


Рисунок 9.14 – Ассемблювання коду

Коди ASCII

Коди *ASCII* наведено на рисунку 9.15, нижче.

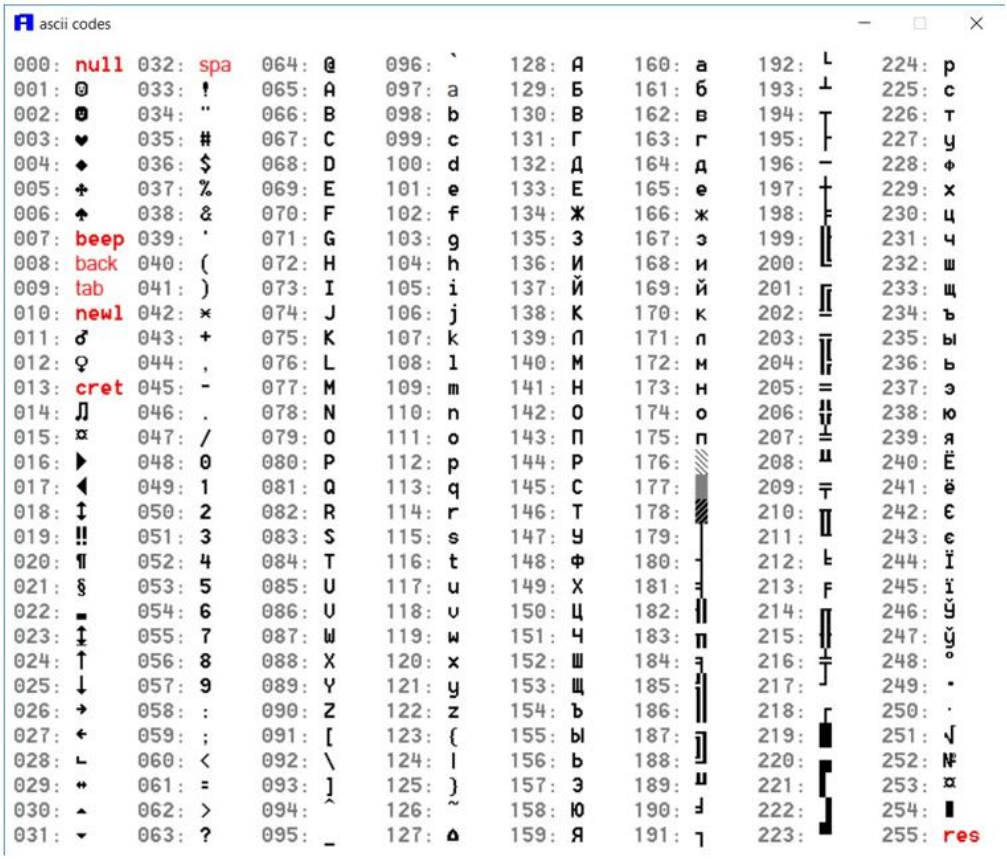


Рисунок 9.15 - Коди *ASCII*

Хід роботи

А. Вивчити структуру програми на мові програмування Асемблер та основні оператори, які використовуються в програмі, поданій в теоретичних відомостях.

Б. З метою кращого розуміння порядку створення та асемблювання програми, розробленої з використанням мови Assembler, написати та виконати програму яка

відображає на екрані монітора прізвище, ім’я і по-батькові студента (латинськими літерами). Кожне слово повинне виводитись в новому рядку.

В. Скомпілювати програму.

Г. Перевірити програму на наявність помилок, у випадку їх наявності — виправити.

Ґ. Скомпонувати програму. Якщо в результаті компонування отриманий виконуваний файл, то виконати програму.

Д. Результатом виконання лабораторної роботи повинна бути програма, яка виводить вкомандній стрічці латинськими літерами прізвище, ім’я і по-батькові студента.

Е. Створити звіт до роботи. Зміст звіту

- Титульна сторінка з темою та метою роботи.
- Завдання до виконання лабораторної роботи.
- Текст (лістинг) програми.
- Скріншот екрану з результатами асемблювання, компонування та виконання програми.
- Висновки.

Є. Надати в звіті відповіді на контрольні запитання нижче.

Контрольні запитання

1. Охарактеризуйте основні сегменти програм та їх призначення.
2. Наведіть особливості синтаксису мови Асемблер.
3. Який спосіб запису коментарів до програми?
4. Поясніть призначення та спосіб визначення міток в програмах на мові Асемблер.
5. Опишіть формат кодування команд на мові Асемблер.
6. Які відмінності між стандартними та спрощеними директивами керування сегментами?
7. Які типи переривань ви знаєте?
8. Як здійснюється визначення даних, необхідних для використання в програмі?
9. Перелічіть основні команди переміщення даних та обмеження на їх використання.

РОБОТА №10. КОМАНДИ ПОРІВНЯННЯ НА АСЕМБЛЕРІ

ТЕМА: Зчитування даних з клавіатури. Команди порівняння. Умовні та безумовні переходи.
МЕТА: Вивчення можливостей зчитування символів з клавіатури, використання команд порівняння і умовних та безумовних переходів в мові програмування Асемблер.

Теоретичні відомості
Приклад програми для зчитування символів з клавіатури

```
.MODEL small
.STACK 4096
.DATA
  Question DB 'ARE YOU A STUDENT? - [Y/N] $'
  GreetingStudent DB 10, 13, 'HELLO STUDENT!$'
  GreetingTeacher DB 10, 13, 'HELLO TEACHER!$'
.CODE
main PROC
mov ax,@Data
mov ds,ax ; встановити регістр DS так, щоб він вказував на сегмент даних
mov dx,OFFSET Question ; посилання на повідомлення-запитання
mov ah,09h ; функція DOS виводу стрічки
int 21h
mov ah,01h ; отримати дані вводу одного символа
int 21h
cmp al,'Y' ; велика буква Y
jz IsStudent ; так, студент
cmp al,'y' ; маленька буква y
jnz IsTeacher ; ні, не студент
IsStudent:
mov dx,OFFSET GreetingStudent ; вказує на привітання "HELLO STUDENT"
jmp DisplayGreeting
IsTeacher:
mov dx,OFFSET GreetingTeacher ; вказує на привітання "HELLO TEACHER"
DisplayGreeting:
mov ah,09h ; функція DOS виводу повідомлення
int 21h ; вивести відповідне повідомлення
mov ah,04ch ; функція DOS завершення програми
int 21h ; завершити програму
main ENDP
END main
```

Таким чином в програму було додано два дуже важливих нових інструменти: можливість введення і прийняття рішень. Ця програма запитує в користувача «ARE YOU A STUDENT?», зчитуючи відповідь (один символ) з клавіатури. Якщо такою відповіддю буде буква «Y» у верхньому або нижньому регістрі, що означає відповідь ТАК («Yes»), то програма виводить повідомлення «HELLO STUDENT!», в іншому випадку виводиться повідомлення «HELLO TEACHER!». У даній програмі є всі основні елементи корисної програми: введення інформації зовнішнього середовища, обробка даних і прийняття рішення.

Функція отримання символу (INT 21h 01H)

Для отримання інформації про код натиснутого на клавіатурі символу потрібно використати функцію переривання з кодом 01h:

AH = 01H
Виведення символу
AL = 8 бітний код числа

Регістр прапорів

Фізичний зміст регістра прапорів такий же, як і у інших регістрів (32 біта).

Eflags Flags

А от призначення - спеціальне. У цей регістр не можна просто записати значення командою MOV. У нього навіть немає імені. Він змінюється по-іншому. І він сильно відрізняється від першого спеціального регістра - EIP (вказівника поточної інструкції).

Регістри загального призначення програміст може сприймати як сховище:

- Числа (адреси або значення).
- Чисел (по тетрадам, байтам, словам тощо).
- Бітової інформації.

Регістр EIP (або IP) можна вважати лише як сховище числа, і воно завжди означає адресу. А ось регістр прапорів – тільки як сховище бітової інформації!

Зрозуміло, що з точки зору комп'ютера будь-яка інформація бітова. Однак з точки зору програміста є числа, які можна ділити, множити і проводити з ними складні обчислення. У цих числах важливіше оперувати цілим байтом, а не кожним окремим бітом.

Існують такі байти, кожен біт яких має самостійне значення: стан процесора, результат порівняння і багато іншого. Звичайно, ці байти теж можуть виглядати як число в шіснадцятковій системі числення, але воно, навпаки, в цілому вигляді не має явного сенсу.

Регістр прапорів - це якраз 16 біт, які розглядаються окремо один від одного.

Регістр прапорів

				O	D	I	T	S	Z		A		P		C
15				11	10	9	8	7	6		4		2		0

Регістр прапорів містить дев'ять найбільш використовуваних прапорів:

CF - прапор перенесення;

PF - прапор парності;

AF - прапор додаткового перенесення;

ZF - прапор нуля;

SF - прапор знаку;

TF - прапор трасування (пастки);

IF - прапор дозволу переривання;

DF - прапор напрямку;

OF - прапор переповнення.

Прапор перенесення (Carry flag, або CF) встановлюється у випадку, якщо при виконанні беззнакової арифметичної операції результатом є число, розрядність якого перевищує розрядність виділеного для нього поля результату. Існують команди безпосереднього встановлення (STC) і скидання (CLC) прапора переносу:

1 - арифметична операція здійснила перенесення з старшого біта результату, старшим є 7-й, 15-й або 31-й біт в залежності від розмірності операнда;

0 - перенесення не було.

Прапор парності (Parity flag, або PF) встановлюється у випадку, якщо в результаті виконання арифметичної або логічної операції результатом є число, що містить парну кількість одиничних бітів у 8-розрядному числі або в молодшому байті 16-розрядного числа. Цей прапор корисний при тестуванні пам'яті і при контролі правильності передачі даних:

1 - 8 молодших розрядів (цей прапор застосовується тільки для 8 молодших розрядів операнда будь-якого розміру) результату містять парне число одиниць;

0 - 8 молодших розрядів результату містять непарне число одиниць.

Прапор додаткового перенесення (Auxiliary Carry, або AF) встановлюється, якщо при виконанні арифметичної операції з 8-розрядним операндом відбувається перенесення з третього біта в четвертий. Застосовується тільки для команд, що працюють з BCD-числами. Фіксує факт позичання з молодшої тетради результату.

Прапор нуля (Zero flag, або ZF) встановлюється, якщо при виконанні арифметичної або логічної операції результатом є число, рівне нулю (тобто всі біти результату рівні 0):

1 - результат нульовий;

0 - результат ненульовий.

Прапор знаку (Sign flag, або SF) встановлюється, якщо при виконанні арифметичної або логічної операції результатом є негативне число (тобто старший біт результату рівний 1). У стандартному додатковому коді одиниця в старшому розряді результату означає отримання негативного числа. Відображає стан старшого біта результату (біти 7, 15 або 31 для 8-, 16- або 32-розрядних операндів відповідно):

1 - старший біт результату дорівнює 1;

0 - старший біт результату дорівнює 0.

Прапор трасування (пастки) (trace flag, або TF) призначений для організації покрокової роботи процесора:

1 - процесор генерує переривання з номером 1 після виконання кожної машинної команди (може використовуватися при відладці програм);

0 - звичайна робота.

Прапор дозволу переривання (interrupt enable flag, або IF) призначений для дозволу або заборони (маскування) апаратних переривань (зовнішніх переривань по входу INTR). Існують команди безпосереднього встановлення (STI) і скидання (CLI) прапора переривання:

1 - апаратні переривання дозволені;

0 - апаратні переривання заборонені.

Прапор напрямку (direction flag, або DF) використовується рядковими командами. Значення прапора DF визначає напрямок поелементної обробки в цих операціях: від початку рядка до кінця (DF = 0) або, навпаки, від кінця рядка до його початку (DF = 1). Для роботи з прапором DF існують спеціальні команди CLD (зняти прапор DF) і CTD (встановити прапор DF). Застосування цих команд дозволяє привести прапор DF у відповідність з алгоритмом і забезпечити автоматичне збільшення або зменшення лічильників при виконанні операцій з рядками.

Прапор переповнення (Overflow flag, або OF) встановлюється у випадку, якщо при виконанні арифметичної операції із знаком результатом є число, розрядність якого перевищує розрядність виділеного для нього поля результату. Використовується для фіксації факту втрати значущого біта при арифметичних операціях:

1 - в результаті операції відбувається перенесення в старший знаковий біт результату або позичання з старшого знакового біта результату (біти 7,15 або 31 для 8-, 16- або 32-розрядних операндів відповідно);

0 - в результаті операції не відбувається перенесення в старший знаковий біт результату або позичання з старшого знакового біта результату.

Команда CMP

Команда CMP походить від англ. слова compare – порівнювати.

Формат: CMP (операнд призначення), (операнд-джерело)

Команда CMP (порівняння) віднімає операнд-джерело з операнда призначення, не змінюючи при цьому значення операндів. Операнди можуть бути байтовими або двобайтовими числами. Хоча значення операндів не змінюються, значення прапорів оновлюються, що може бути враховано в наступних командах умовного переходу. Команда CMP впливає на прапори AF, CF,

OF, PF, SF і ZF. При збігу значень операндів встановлюється прапор ZF. Прапор перенесення встановлюється, якщо операнд призначення менший ніж операнд-джерело.

В приведеному прикладі важливий лише прапор ZF.

Наприклад, щодо прапора нуля (ZF) дія команди виглядає так: MOV AH, 10h; для наочності присвоїмо AH значення 10h

CMP AH, 8 ; ZF = 0

CMP AH, 10h ; ZF = 1

CMP AH, 0A0h ; ZF = 0

1.5 Команда умовного переходу JNZ (JNE)

Команда умовного переходу JNZ (JNE) походить від англ. слів Jmp if Not Zero - стрибнути якщо Zero Flag (ZF) не встановлений, тобто не рівний одиниці (тоді коли ZF = 0).

Формат:

JNZ мітка

Дія Якщо ZF = 0, то дія схожа на команду JMP (зміна EIP на вказану адресу, тобто перехід куди сказали).

Якщо ZF = 1, то управління переходить до наступної команди (зміна EIP на адресу наступної команди, тобто нічого не відбувається).

JZ (JE) - команда протилежної дії.

1.6 Умовні та безумовні переходи

Безумовний перехід - це перехід, який виконується завжди. Безумовний перехід здійснюється за допомогою команди JMP. У цієї команди є один операнд, який може бути безпосередньою адресою (міткою), регістром або осередком пам'яті, яка містить адресу. Існують також «далекі» переходи - між сегментами.

Мітка – це набір з символів латинського алфавіту, цифр та/або символу нижнього підкреслення («_») який починається з символу і закінчується двокрапкою («:»).

JMP mitka ; Перехід на мітку

JMP bx ; Перехід за адресою в BX

JMP word [bx] ; Перехід за адресою, що міститься в пам'яті за адресою в BX

Умовний перехід здійснюється, якщо виконується певна умова, задана прапорами процесора (крім однієї команди, яка перевіряє CX на рівність нулю). Стан прапорів змінюється після виконання арифметичних, логічних і деяких інших команд. Якщо умова не виконується, то управління переходить до наступної команди.

Існує багато команд для різних умовних переходів. Також для деяких команд є синоніми (наприклад, JZ і JE - це одне й те саме). Для наочності всі команди умовних переходів наведені в табл. 10,1, а значення абревіатур в назві команди jxx в табл. 10,2:

Таблиця 10.1 – Команди умовних переходів

Типи операндів	Команда	Критерій умовного переходу	Значення прапорів для переходу
Будь-які	JZ / JE	операнд_1 = операнд_2	ZF=1
Будь-які	JNZ / INE	операнд_1 ≠ операнд_2	ZF=0
Числа зі знаком	JL / JNGE	операнд_1 < операнд_2	SF ≠ OF
Числа зі знаком	JLE / JNG	операнд_1 <= операнд_2	SF ≠ OF або ZF = 1
Числа зі знаком	JG / JNLE	операнд_1 > операнд_2	SF = OF і ZF = 0
Числа зі знаком	JGE / JNL	операнд_1 => операнд_2	SF = OF
Числа без знаку	JB / JNAE / JC	операнд_1 < операнд_2	CF = 1
Числа без знаку	JBE / JNA	операнд_1 <= операнд_2	CF = 1 або ZF = 1
Числа без знаку	JA / JNBE	операнд_1 > операнд_2	CF = 0 і ZF = 0
Числа без знаку	JAE / JNB / JNC	операнд_1 => операнд_2	CF = 0
Будь які	JP	число одиничних бітів парне	PF = 1
Будь які	JNP	число одиничних бітів непарне	PF = 0
Числа зі знаком	JS	знак дорівнює 1	SF = 1
Числа зі знаком	JNS	знак дорівнює 0	SF = 0
Числа без знаку	JO	є переповнення	OF = 1
Числа без знаку	JNO	немає переповнення	SF = 1
Будь які	JCXZ	вміст CX дорівнює нулю	SF = 0

Таблиця 10.2 – Значення абревіатур в назві команди jxx

Позначення	Оригінальний термін	Переклад	Тип операндів
e	Equal	Рівно	Будь які
n	Not	Ні	Будь які
g	Greater	Більше	Числа зі знаком
l	Less	Менше	Числа зі знаком
a	Above	Вище (в значенні більше)	Числа без знаку
b	Below	Нижче (в значенні менше)	Числа без знаку

У всіх цих команд один операнд - ім'я мітки для переходу. Зверніть увагу, що деякі команди застосовуються для беззнакових чисел, а інші - для чисел із знаком. Порівняння «вище» і «нижче» відносяться до беззнакових чисел, а «більше» і «менше» - до чисел із знаком. Для беззнакових чисел ознакою переповнення буде прапор CF, а відповідними командами переходу JC і JNC. Для чисел із знаком про переповнення можна судити за станом прапора OF, тому їм відповідають команди переходу JO і JNO. Команди переходів не змінюють значення прапорів.

Хід роботи

1. Вивчити основні поняття та оператори, які використовуються в програмі, поданій в теоретичних відомостях.

2. Ввімкнути ЕОМ.

3. Написати програму яка виконує завдання відповідно до варіанту (табл.3). В програмі використовувати символи латинського алфавіту. При зчитуванні з клавіатури використовувати як маленькі так і великі символи.

Таблиця 3 – Варіанти завдань

№ варіанту	Завдання
1	Програма пропонує вам ввести символи ‘N’ або ‘S’. При введенні символу ‘N’ – виводить на екран ваше ім’я. При введені символу ‘S’ – виводить ваше прізвище. Після цього програма завершується. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
2	Програма пропонує вам ввести порядковий номер дня тижня. Після його введення виводить на екран повну назву відповідного дня і завершує програму. У випадку введення символу, який не відповідає жодному з днів тижня – виводить повідомлення про помилку і пропонує ввести порядковий номер ще раз.
3	Програма пропонує вам ввести символи ‘N’ або ‘F’. При введенні символу ‘N’ – виводить на екран ваше ім’я. При введені символу ‘F’ – виводить по-батькові. Після цього програма завершується. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
4	Програма пропонує вам ввести порядковий номер календарного місяця (від 1 до 9). Після його введення виводить на екран повну назву відповідного місяця і завершує програму. У випадку введення символу, який не відповідає жодному з місяців – виводить повідомлення про помилку і пропонує ввести порядковий номер ще раз.
5	Програма запитує вас чи ви чоловічої статі чи жіночої. При введенні символу ‘M’ - виводить повідомлення ‘MAN’ і ваше ім’я, при введені символу ‘W’ – виводить повідомлення ‘WOMAN’ і ваше ім’я. Після цього програма завершується. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
6	Програма запитує вас який зараз час доби. При введенні символу ‘M’ - виводить привітання ‘GOOD MORNING’ і ваше ім’я, при введені символу ‘D’ – виводить повідомлення ‘GOOD DAY’ і ваше ім’я, при введенні символу ‘E’ - виводить привітання ‘GOOD EVENING’ і ваше ім’я, при введені символу ‘N’ – виводить повідомлення ‘GOOD NIGHT’ і ваше ім’я. Після цього програма завершується. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ

	ще раз.
7	Програма пропонує вам ввести символи ‘N’, ‘S’ або ‘F’. При введенні символу ‘N’ – виводить на екран ваше ім’я. При введені символу ‘S’ – виводить ваше прізвище. При введені символу ‘F’ – виводить по-батькові. Після цього програма знову пропонує ввести один з цих символів. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
8	Програма запитує вас чи ви чоловічої статі чи жіночої. При введенні символу ‘M’ - виводить повідомлення ‘MAN’ і ваше по-батькові, при введені символу ‘W’ – виводить повідомлення ‘WOMAN’ і ваше по-батькові. Після цього програма завершується. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
9	Програма пропонує вам ввести символи ‘I’ або ‘P’. При введенні символу ‘I’ – виводить на екран ваше ім’я. При введені символу ‘P’ – виводить ваше прізвище. Після цього програма завершується. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
10	Програма пропонує вам ввести символи ‘N’ або ‘F’. При введенні символу ‘N’ – виводить на екран ваше ім’я. При введені символу ‘F’ – виводить по-батькові. Після цього програма знову пропонує ввести один з цих символів. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
11	Програма запитує вас який зараз час доби. При введенні символу ‘M’ - виводить привітання ‘GOOD MORNING’ і ваше ім’я, при введені символу ‘D’ – виводить повідомлення ‘GOOD DAY’ і ваше ім’я, при введенні символу ‘E’ - виводить привітання ‘GOOD EVENING’ і ваше ім’я, при введені символу ‘N’ – виводить повідомлення ‘GOOD NIGHT’ і ваше ім’я. Після цього програма знову пропонує ввести один з цих символів. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
12	Програма пропонує вам ввести символи ‘I’ або ‘P’. При введенні символу ‘I’ – виводить на екран ваше ім’я. При введені символу ‘P’ – виводить ваше прізвище. Після цього програма знову пропонує ввести один з цих символів. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
13	Програма запитує вас чи ви чоловічої статі чи жіночої. При введенні символу ‘M’ - виводить повідомлення ‘MAN’ і ваше прізвище, при введені символу ‘W’ – виводить повідомлення ‘WOMAN’ і ваше прізвище. Після цього програма завершується. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
14	Програма запитує вас який зараз час доби. При введенні символу ‘M’ - виводить привітання ‘GOOD MORNING’ і ваше прізвище, при введені

	символу ‘D’ – виводить повідомлення ‘GOOD DAY’ і ваше прізвище, при введенні символу ‘E’ - виводить привітання ‘GOOD EVENING’ і ваше прізвище, при введені символу ‘N’ – виводить повідомлення ‘GOOD NIGHT’ і ваше прізвище. Після цього програма завершується. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
15	Програма пропонує вам ввести символи ‘I’ або ‘P’. При введенні символу ‘I’ – виводить на екран ваше ім’я. При введені символу ‘P’ – виводить ваше прізвище. Після цього програма знову пропонує ввести один з цих символів. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
16	Програма пропонує вам ввести порядковий номер дня тижня. Після його введення виводить на екран повну назву відповідного дня. Після цього програма знову пропонує ввести порядковий номер дня тижня. У випадку введення символу, який не відповідає жодному з днів тижня – виводить повідомлення про помилку і пропонує ввести порядковий номер ще раз.
17	Програма пропонує вам ввести символи ‘N’ або ‘S’. При введенні символу ‘N’ – виводить на екран ваше ім’я. При введені символу ‘S’ – виводить ваше прізвище. Після цього програма знову пропонує ввести один з цих символів. У випадку введення іншого символу – виводить повідомлення про помилку і пропонує ввести символ ще раз.
18	Програма пропонує вам ввести порядковий номер календарного місяця (від 1 до 9). Після його введення виводить на екран повну назву відповідного місяця. Після цього програма знову пропонує ввести порядковий номер календарного місяця. У випадку введення символу, який не відповідає жодному з місяців – виводить повідомлення про помилку і пропонує ввести порядковий номер ще раз.

4. Проасемблювати програму.

5. Перевірити програму на наявність помилок, у випадку їх наявності — виправити і перейти до кроку 3.

6. Скомпонувати програму.

7. Якщо в результаті компонування отриманий виконуваний файл, то виконати програму.

8. Результатом виконання лабораторної роботи повинна бути програма, яка виводить в командній стрічці повідомлення відповідно до завдання поданого в п. 3.

Зміст звіту

1. Титульна сторінка, оформлена відповідно до зразка.
2. Тема роботи
2. Мета роботи.
3. Завдання до виконання лабораторної роботи.
7. Текст (лістинг) програми.
8. Скріншот екрану з результатами асемблювання, компонування та виконання програми.
9. Письмові відповіді на контрольні запитання.
10. Висновки.
11. Список джерел інформації.

Контрольні запитання

1. Які ви знаєте команди умовного переходу в мові програмування Асемблер?
2. Що таке безумовний перехід в мові програмування Асемблер?

3. Що таке мітка в мові програмування Асемблер?
4. Як можна зчитувати символ з клавіатури?
5. Як виводити рядок на екран монітора в мові програмування Асемблер?
6. Наведіть приклад використання команди порівняння.

РОБОТА №11. АРИФМЕТИЧНІ ОПЕРАЦІЇ В МОВІ ПРОГРАМУВАННЯ ASSEMBLER

МЕТА: Вивчити основні команди мови програмування Асемблер для здійснення арифметичних операцій та навчитись використовувати їх в процесі написання програм.

Теоретичні відомості

Розгляньте код програми, що наведений на лістингу 11.1. Дана програма зчитує число X з клавіатури, виконує обчислення за формулою: *Y = (X+1)*2*. Після цього виводить результат обчислення – число Y на екран.

Лістинг 11.1 – Приклад програми обчислення

```
.MODEL small
.STACK 4096
.DATA
    Set_X DB 'X = $'
    Result DB 13,10,'Y = $'
    error_ db "incorrect number$"
    buff db 6,7 Dup(?)
.CODE
main PROC
    mov ax,@Data
    mov ds,ax ; встановити регістр DS таким чином, щоб він вказував на
сегмент даних
    mov dx,OFFSET Set_X ; посилання на повідомлення-запитання
    mov ah,09h ; функція DOS виводу стрічки
    int 21h
;*************************************************************************
;*Зчитування числа з клавіатури і перетворення його в цифрове значення
;******** Результат в регістрі AX ********
;*************************************************************************
    mov cx,0
    mov ah,0ah
    xor di,di
    mov dx,offset buff ; адреса буфера
    int 21h ; зчитуємо стрічку
    mov dl,0ah
    mov ah,02
    int 21h ; виводимо перевід рядка
    ; обробляємо вміст буфера
    mov si,offset buff+2 ; берем адресу початку стрічки
    cmp byte ptr [si],"-" ; якщо перший символ мінус
    jnz ii1
    mov di,1 ; встановлюємо прапор
    inc si ; і пропускаєм його
ii1:
```

```
    xor ax,ax
    mov bx,10
ii2:
    mov cl,[si] ; берем символ з буфера
    cmp cl,0dh ; перевіряємо чи він не останній
jz endin:
; якщо символ не останній, то перевіряємо його на правильність
    cmp cl,'0' ; якщо введений невірний символ <0
    jl er
    cmp cl,'9' ; якщо введений невірний символ >9
    ja er
    sub cl,'0' ; робимо з символу число
    mul bx ; перемножуємо на 10
    add ax,cx ; додаємо до інших
    inc si ; вказівник на наступний символ
    jmp ii2 ; повторюємо
er: ; якщо була помилка, то виводимо повідомлення про це і виходимо
    mov dx, offset error_
    mov ah,09
    int 21h
    int 20h
    ; все символи з буфера оброблені число знаходиться в ax
endin:
    cmp di,1 ; якщо встановлений прапорець, то
    jnz ii3
    neg ax ; робим число від'ємним
ii3:
;*************************************************************************
;**Завершення функції зчитування і розпізнавання числа ***************
;*************************************************************************
    add ax,1
    mov bx,2
    mul bx
    xchg cx,ax
    mov dx,OFFSET Result ; вказує на
    mov ah,09h ; функція DOS виводу повідомлення
    int 21h ; вивести відповідне повідомлення
    xchg cx,ax
;*************************************************************************
;** Початок функції перетворення числа в рядок і виводу її на екран **
;*************************************************************************
; Перевіряємо число на знак
    test ax, ax
    jns oi1
; Якщо воно від'ємне, виведемо мінус і залишимо його модуль.
    mov cx, ax
    mov ah, 02h
    mov dl, '-'
```

```
    int 21h
    mov ax, cx
    neg ax
; Кількість цифр будемо тримати в CX.
oi1:
    xor cx, cx
    mov bx, 10 ; основа сс. 10 для десяткової і т.п.
oi2:
    xor dx,dx
    div bx
; Ділимо число на основу сс. В залишку виходить остання цифра.
; Відразу виводити її неможна, тому збережемо її в стек.
    push dx
    inc cx
; А з остачею повторюємо те ж саме, відділяючи від нього чергову
; цифру справа, поки не залишиться нуль, що означає, що далі
; зліва лише нулі.
    test ax, ax
    jnz oi2
; Тепер приступимо до виводу.
    mov ah, 02h
oi3:
    pop dx
; Беремо чергову цифру, переводимо її в символ и виводимо.
    add dl, '0'
    int 21h
; Повторимо рівно стільки разів, скільки цифр нарахували.
    loop oi3
;*************************************************************
;* Завершення функції перетворення числа в рядок і виводу її на екран *
;*************************************************************
    mov ah,04ch ; функція DOS завершення програми
    int 21h ; завершити програму
main ENDP
END main
```

Таблиця 11.1 – Команди арифметичних операцій мови Асемблер

Додавання	
ADD	Додавання байта або слова
ADC	Додавання байта або слова з розрядом переносу
INC	Збільшення байта або слова на 1
AAA	Корекція додавання неупакованих десяткових чисел
DAA	Корекція додавання упакованих десяткових чисел
Віднімання	
SUB	Віднімання байта або слова
SBB	Віднімання байта або слова з розрядом переносу
DEC	Зменшення байта або слова на 1
NEG	Інверсія байта або слова
CMP	Порівняння байта або слова

AAS	Корекція віднімання неупакованих десяткових чисел
DAS	Корекція віднімання упакованих десяткових чисел
Множення	
MUL	Множення беззнакового байта або слова
IMUL	Цілочисельне множення байта або слова
AAM	Корекція множення неупакованих десяткових чисел
Ділення	
DIV	Ділення беззнакового байта або слова
IDIV	Цілочисельне ділення байта або слова
AAD	Корекція ділення неупакованих десяткових чисел
CWB	Перетворення байта в слово
CWD	Перетворення слова в подвійне слово

Команди додавання:

ADD (операнд призначення), (операнд-джерело)

Сума двох операндів, які можуть бути байтами або словами, поміщається в операнд призначення. Обидва операнда можуть бути знаковими або беззнаковими числами. Команда ADD змінює значення прапорів AF, CF, OF, PF, SF і ZF.

Команда **ADC** (додавання з врахуванням розряду переносу) додає операнди, які можуть бути байтами або словами, і додає 1, якщо встановлено розряд переносу, результат поміщається в операнд призначення. Обидва операнда можуть бути знаковими або беззнаковими числами. Команда ADD змінює значення прапорів AF, CF, OF, PF, SF і ZF. Команда ADC враховує значення розряду переносу від попередньої операції, це може бути використано для організації додавання чисел довільної розрядності.

Команда **INC** (інкремент) додає одиницю до операнду призначення. Операнд може бути байтом або словом і трактується як беззнакове двійкове число. Команда INC змінює значення прапорів AF, OF, PF, SF і ZF значення прапора CF ця команда не змінює.

Команда **AAA** (корекція додавання неупакованих десяткових чисел) приводить вміст регістра AL до виду правильного неупакованого десяткового числа, старший напівбайт при цьому обнуляється. Команда AAA змінює значення прапорів FC і AC вміст прапорів OF, PF, SF і ZF після виконання команди AAA невизначено.

Команда **DAA** (десяткова корекція додавання) приводить вміст регістра AL до виду правильного упакованого десяткового числа після попередньої команди додавання. Команда DAA змінює значення прапорів AF, CF, PF, SF і ZF вміст прапора OF після виконання команди DAA не визначено.

Для додавання двох чисел призначена команда ADD. Вона працює як з числами зі знаком, так і з числами без знаку (це особливість додаткового коду).

Операнди повинні мати однаковий розмір (не можна додавати 16- і 8-бітове значення). Результат поміщається на місце першого операнда. Загалом, ці правила справедливі для більшості команд.

Після виконання команди змінюються прапори, за якими можна визначити характеристики результату:

Прапор CF встановлюється, якщо при додаванні відбулося перенесення із старшого розряду. Для беззнакових чисел це означатиме, що відбулося переповнення і результат вийшов некоректним.

Прапор OF позначає переповнення для чисел із знаком.

Прапор SF дорівнює знаковому біту результату (для чисел зі знаком, а для беззнаковихвін дорівнює старшому біту і особливо сенсу не має).

Прапор ZF встановлюється, якщо результат дорівнює 0.

Прапор PF – ознака парності, дорівнює 1, якщо результат містить непарне число одиниць.

Приклади:

```
add ax, 5   ; AX = AX + 5
add dx, cx  ; DX = DX + CX
add dx, cl  ; Помилка: різний розмір операндів.
```

Команди віднімання

SUB (операнд призначення), (операнд-джерело)

Вміст операнда-джерела віднімається з вмісту операнда призначення, і результат поміщається в операнд призначення. Операнди можуть бути знаковими або беззнаковими, двійковими або десятковими (див. команди AAS і DAS), однобайтовими або двобайтовими числами. Команда SUB змінює значення прапорів AF, CF, OF, PF, SF і ZF.

Насправді віднімання в процесорі реалізовано за допомогою додавання. Процесор змінює знак другого операнда на протилежний, а потім додає два числа. Якщо вам необхідно в програмі поміняти знак числа на протилежний, можна використовувати команду NEG. У цієї команди всього один операнд.

Приклад:

```
sub si, dx  ; SI = SI - DX
neg ax  ; AX = -AX
```

SBB (операнд призначення), (операнд-джерело)

Команда SBB (віднімання з урахуванням позичання) віднімає вміст операнда-джерела від вмісту операнда призначення, потім віднімає від результату 1, якщо був встановлений прапор перенесення CF. Результат поміщається на місце операнда призначення.

Операнди можуть бути знаковими або беззнаковими, двійковими або десятковими (див. команди AAS і DAS), однобайтовими або двобайтовими числами. Команда SBB змінює значення прапорів AF, CF, OF, PF, SF і ZF. Команда SBB може бути використана для здійснення віднімання мультибайтних чисел.

Команда **DEC** (декремент) віднімає одиницю з операнда призначення, який може бути одно- або двобайтовим. Команда DEC змінює вміст прапорів AF, OF, PF, SF і ZF. Вміст прапора CF при цьому не змінюється.

Команда **NEG** (інверсія) віднімає операнд призначення, який може бути байтом або словом і поміщає результат в операнд призначення. Така форма двійкового доповнення числа придатна для інверсії знакових цілих чисел. Якщо операнд нульовий, його знак не змінюється. Спроба застосувати команду NEG до байтового числа - 128 або до двобайтового числа - 32768 не призводить до зміни значення операнда, але встановлює прапор OF.

Команда NEG впливає на прапори AF, CF, OF, PF, SF і ZF. Прапор CF завжди встановлений за винятком випадку, коли операнд дорівнює нулю, коли цей прапор скинутий в стан логічного «0».

Приклад:

```
neg bx    ; BX = -BX
```

Команда **CMP** (порівняння) віднімає операнд-джерело з операнда призначення, не змінюючи при цьому значення операндів. Операнди можуть бути байтовими або двобайтовими числами. Хоча значення операндів на змінюються, значення прапорів оновлюються, що може бути враховано в наступних командах умовного переходу. Команда CMP впливає на прапори AF, CF, OF, PF, SF і ZF. При збігу значень операндів

встановлюється прапор ZF. Прапор перенесення встановлюється, якщо операнд призначення менший за операнд-джерело.

Команда **AAS** (корекція віднімання неупакованих десяткових чисел) коригує результат попереднього вирахування двох правильних упакованих десяткових чисел. Операндом призначення в команді віднімання повинен бути регістр AL. Команда AAS призводить значення в AL до виду правильного неупакованого десяткового числа; старший напівбайтпри цьому обнуляється. AAS впливає на прапори AF і CF. Значення прапорів OF, PF, SF і ZF після виконання команди AAS невизначено.

Команда **DAS** (десяткова корекція вирахування) коригує результат попереднього вирахування двох правильних упакованих десяткових чисел. Операндом призначення в команді віднімання повинен бути регістр AL. Команда DAS призводить значення в AL до виду двох правильних упакованих десяткових чисел. Команда DAS впливає на прапори AF і CF. Значення прапорів OF, PF, SF і ZF після виконання команди DAS невизначено.

Команди множення

Множення чисел без знакуMUL (опранд-джерело)

Команда MUL (множення) виконує беззнакове множення операнда-джерела і вмісту регістра AX (AL). У цієї команди тільки один операнд - другий множник, який повинен знаходитися в регістрі або в пам'яті. Розташування першого множника і результату задається неявно і залежить від розміру операнда. Якщо операнд-джерело однобайтовий, здійснюється множення на вміст регістра AL, а двобайтовий результат повертається регістри AH і AL. Якщо операнд-джерело двобайтовий, здійснюється множення на вміст регістра AX, а чотирьохбайтовий результат повертається в регістри DX і AX:

Відмінність множення від додавання і віднімання в тому, що розрядність результату виходить в 2 рази більшою, ніж розрядність співмножників. Також і в десятковій системі - наприклад, перемножуючи двозначне число на двозначне, ми можемо отримати в результаті максимум чотиризначне. Запис «DX:AX» означає, що старше слово результату буде знаходитися в DX, а молодше - в AX. Приклади:

```
mul bl; AX = AL * BL
mul ax; DX: AX = AX * AX
```

Операнди розглядаються як беззнакові двійкові числа. Якщо старша половина результату (регістр AH при однобайтовому множенні і DX при двобайтовому множенні) встановлюються прапори CF і OF, в іншому випадку ці прапори скидаються (тобто встановлюються в нульове значення). У цьому випадку старшу частину результату можна відкинути. Цю властивість можна використовувати в програмі, якщо результат повинен бути такого ж розміру, що і множники.

Якщо після виконання множення встановлені прапори CF і OF, це говорить про наявність значущих цифр результату в регістрі AH або DX. Вміст прапорів AF, PF SF і ZF після виконання команди множення невизначено.

Множення чисел зі знаком

Команда **IMUL** (цілочисельне множення зі знаком) виконує знакове множення операнда-джерела і вмісту регістра AX (AL). Якщо операнд-джерело однобайтовий, здійснюється множення на вміст регістра AL, а двобайтовий результат повертається в регістри AH і AL. Якщо операнд-джерело двобайтовий, здійснюється множення на вміст регістра AX, а чотирьохбайтовий результат повертається регістри DX і AX. Операнди розглядаються як беззнакові двійкові числа. Якщо старша половина результату (регістр AH при однобайтовому

множенні і DX при двобайтовому множенні) встановлюються прапори CF і OF, в іншому випадку ці прапори додаються. Якщо після виконання множення встановлені прапори CF і OF, це говорить про наявність значущих цифр результату в регістрі AH або DX. Вміст прапорів AF, PF SF і ZF після виконання команди цілочисельного множення невизначено.

Ця команда має три форми, що розрізняються кількістю операндів:

• *З одним операндом* - форма, аналогічна команді MUL. Як операнд вказується множник. Розташування іншого множника і результату визначається за таблицею.

• *З двома операндами* - вказуються два множника. Результат записується на місце першого множника. Старша частина результату в цьому випадку ігнорується. До речі, ця форма команди не працює з операндами розміром 1 байт.

• *З трьома операндами* - вказується положення результату, першого і другого множника. Другий множник повинен бути безпосереднім значенням. Результат має такий же розмір, як перший множник, старша частина результату ігнорується. Це форма теж не працює з однобайтними множниками.

Приклади:

```
imul cl; AX = AL * CL
imul si; DX: AX = AX * SI
imul bx, ax; BX = BX * AX
imul cx, - 5; CX = CX * (-5)
```

CF = *OF* = 0, якщо множник поміщається в молодшій половині результату, інакше *CF* = *OF* = 1. Для другої і третьої форми команди *CF* = *OF* = 1 означає, що відбулось переповнення.

Команда **AAM** (корекція множення неупакованих десяткових чисел) призводить результат попереднього множення до двох правильно упакованих десяткових чисел. Для отримання правильного результату після виконання корекції старші напівбайти операндів які перемножуються повинні бути нульовими, а молодші повинні бути правильними двійково-десятковими цифрами.

Команда AAM впливає на прапори PF, SF і ZF. Вміст прапорів AF, CF і OF після виконання команди AAM невизначено.

Команди ділення

Ділення чисел без знаку

Команда **DIV** (ділення) виконує беззнакове ділення вмісту акумулятора (і його розширення) на операнд-джерело. Ділення цілих двійкових чисел - це завжди ділення з залишком! Якщо операнд-джерело однобайтовий, здійснюється ділення двобайтового діленого, розташованого в регістрах AH і AL. Однобайтовий результат ділення зберігається в регістрі AL, а однобайтовий залишок - в регістрі AH. Якщо операнд-джерело двобайтовий, здійснюється ділення чотирибайтового діленого, розташованого в регістрах DX і AX. Двобайтовий результат ділення при цьому зберігається в регістрі AX, а двобайтовий залишок – регістрі DX.

Якщо значення результату ділення перевищує розрядність акумулятора (0FFh для однобайтового ділення і 0FFFFh - для двобайтового) або здійснюється спроба ділення на нуль, генерується переривання типу 0, а результат ділення і залишок залишаються невизначеними. Вміст прапорів AF, CF, OF, PF, SF і ZF після виконання команди DIVневизначено.

Приклади:

```
div cl; AL = AX / CL, залишок у AH
div di; AX = DX: AX / DI, залишок в DX
```

Ділення чисел зі знаком

Команда **IDIV** (цілочисельне ділення зі знаком) виконує знакове ділення вмісту акумулятора (і його розширення) на операнд-джерело. Якщо операнд-джерело однобайтовий, здійснюється ділення двобайтового діленого, розташованого в регістрах AH і AL. Однобайтовий результат ділення розміщується в регістрі AL, а однобайтовий залишок - в регістрі AH. Для однобайтового цілочисельного ділення додатнє ділене не може бути більше за значення +127 (7Fh), а негативне не може бути меншим за -127 (81h). Якщо операнд-джерело двобайтовий, здійснюється ділення чотирибайтового діленого, розміщеного в регістрах DX і AX. Двобайтовий результат ділення при цьому розміщується в регістрі AX, а двобайтовий залишок - в регістрі DX. Для двобайтового цілочисельного ділення додатнє ділене не може бути більше значення +32767 (7FFFh), а негативне не може бути меншим за значення -32767 (8001H).

Якщо ділене позитивне і перевищує максимум або негативне і менше мінімуму, генерується переривання типу 0, а ділене і залишок залишаються невизначеними. Окремим випадком такої події є спроба ділення на нуль. Вміст прапорів AF, CF, OF, PF, SF і ZF після виконання команди IDIV невизначено.

Команда **AAD** (корекція ділення неупакованих десяткових чисел) модифікує вміст регістра AL перед виконанням ділення так, щоб при виконанні ділення в діленому вийшло правильне неупаковане десяткове число. Для отримання правильного результату після виконання ділення вміст регістра AH має бути нульовим. Команда AAD впливає на прапори PF, SF і ZF. Вміст прапорів AF, CF і OF після виконання команди AAD невизначено.

Команда **CBW** (перетворення байта в слово) розширює знак байта в регістрі AL на весь регістр AX. Команда CBW не вплине на прапори. Команда CBW може бути використана для отримання двобайтового діленого з однобайтового перед виконанням команди ділення.

Команда **CWD** (перетворення слова в подвійне слово) розширює знак слова в регістрі AX на пару регістрів AX і DX. CWD Команда не впливає на прапори. Команда CWD може бути використана для отримання чотирибайтового діленого з двобайтового перед виконанням команди ділення.

Команда **XCHG**. Однією із задач в мові програмування Асемблер може бути задача поміняти місцями значення в регістрах. Вирішити це завдання можна використовуючи оператори MOV та додаткову комірку пам'яті, ось так:

```
mov ax, 20
mov cx, 30

mov bx, ax
mov ax, cx
mov cx, bx
```

Виходить, що потрібно виконати три команди. Команда XCHG дозволяє зробити обмін однією командою. Обмін може здійснюватися між регістрами або регістрами і коміркою пам'яті.

Наприклад:

```
mov ax, 20
mov cx, 30
```

```
XCHG ax, cx; поміняти місцями
```

Використовувати цю команду процесора дуже зручно, оскільки це дозволяє оптимізувати операції пов'язані з необхідністю великої кількості замін. Наприклад, при сортуванні. Реально зменшується і розмір програми.

Хід роботи

1. Вивчити основні поняття та команди, які використовуються в програмі, поданій в теоретичних відомостях.
2. Ввімкнути ЕОМ.
3. Написати програму яка просить користувача ввести число X. Потім програма обчислює формулу відповідно до варіанту і видає результат обчислення на екран. Після цього програма запитує користувача чи він хоче завершити програму і пропонує ввести символи ‘Y’ (‘y’) або ‘N’ (‘n’). Якщо користувач вводить символ ‘Y’ (‘y’) – програма завершується, а якщо користувач вводить символ N’ (‘n’), то програма переходить на початок, тобто знову пропонує ввести число X, після чого знову видає результат обчислення.

№ варіанту	Завдання
1	$Y=\frac{(X+3)^2}{2}-1$
2	$Y=\frac{2X-5}{4}-3$
3	$Y=\frac{(3X+4)^2-5}{2}$
4	$Y=\frac{X^3-2}{3}+4$
5	$Y=\frac{(X+2)^2}{10}-3$
6	$Y=\frac{2X-4}{5}+2$
7	$Y=\left(\frac{2-X}{5}+3\right)^2$
8	$Y=\frac{2X-4}{3}+4$
9	$Y=\left(\frac{4-X}{5}\right)^3+1$
10	$Y=\frac{125}{X+4}-1$
11	$Y=\frac{101}{5-X}+4$
12	$Y=\frac{95}{3+X^2}-4$

13	$Y = \frac{88}{(4-X)^2} + 8$
14	$Y = \frac{200}{X^3 + 1} - 4$
15	$Y = \frac{115}{X^2 - 3} + 1$
16	$Y = \frac{62}{3 - X^3} + 2$
17	$Y = \frac{50}{5 - X^4} + 8$
18	$Y = \left(\frac{4}{1+X}\right)^2 - 5$

4. Проасемблювати програму.

5. Перевірити програму на наявність помилок, у випадку їх наявності — виправити іперейти до кроку 3.

6. Скомпонувати програму.

7. Якщо в результаті компонування отриманий виконуваний файл, то виконати програму.

8. Результатом виконання лабораторної роботи повинна бути програма, яка виводить вкомандній стрічці результат обчислення формули, яка задана в п. 3.

Зміст звіту

1. Титульна сторінка, оформлена відповідно до зразка.
2. Тема роботи
2. Мета роботи.
3. Завдання до виконання лабораторної роботи.
7. Текст (лістинг) програми.
8. Скріншот екрану з результатами асемблювання, компонування та виконання програми.
9. Висновки.

Контрольні запитання

1. Які ви знаєте команди додавання в мові програмування Асемблер?
2. Які ви знаєте команди віднімання в мові програмування Асемблер?
3. Які ви знаєте команди ділення в мові програмування Асемблер?
4. Які ви знаєте команди множення в мові програмування Асемблер?
5. Опишіть принцип дії команди ADD.
6. Опишіть принцип дії команди INC.
7. Опишіть принцип дії команди SUB.
8. Опишіть принцип дії команди DEC.
9. Опишіть принцип дії команди NEG.
10. Опишіть принцип дії команди MUL.
11. Опишіть принцип дії команди IMUL.
12. Опишіть принцип дії команди DIV.
13. Опишіть принцип дії команди IDIV.

РОБОТА №12. ВИКОРИСТАННЯ ЗМІННИХ, СТЕКУ ТА ПІДПРОГРАМ В МОВІ ПРОГРАМУВАННЯ АСЕМБЛЕР

МЕТА: Вивчити правила оголошення та використання змінних в мові програмуванняАсемблер та навчитись застосовувати стек та підпрограми.

Теоретичні відомості

Змінні в програмі на асемблері

Змінна - це місце в пам'яті яке має ім'я і тип. Створюючи програму на асемблері нам потрібно здійснювати визначення наших змінних та констант. Змінні зазвичай визначаються в сегменті даних. Асемблер підтримує ряд директив які служать для виділення місця воперативній пам’яті для змінних. Наприклад:

db - 1 байт (8 біт)

dw - 2 байти (16 біт)

dd - 4 байти (32 біт)

dq – 8 байт (64 біти)

```
.MODEL small
.STACK 4096
.DATA
    Data1 DB 31h ; виділити один байт з вмістом 31h
.CODE
main PROC
    mov ax, @data ; встановлення в ds адреси
    mov ds, ax ; сегменту даних
    mov ax, OFFSET Data1 ; де знаходитися змінна
Exit:
    mov ah, 04Ch ; функція DOS виходу з програми
    mov al, 0h ; код повернення
    int 21h ; Виклик DOS зупинка програми
main ENDP
END main
```

Так само цю змінну можна поміщати в регістр ось так наприклад:

```
main PROC
    mov ax, @data ; встановлення в ds адреса
    mov ds, ax ; сегменту даних
    mov ax, OFFSET Data1 ; де знаходитися змінна
    mov al, Data1 ; поміщаємо в регістр AX (AL)
Exit:
```

Можна і зворотну операцію здійснити - помістити дані з регістра в змінну:

```
main PROC
```

```
    mov ax, @data ; встановлення в ds адреси
    mov ds, ax ; сегменту даних
    mov ax, OFFSET Data1 ; де знаходитися змінна
    mov al, Data1 ; поміщаємо в регістр AX (AL)
    inc al ; збільшити на одиницю
    mov Data1, al ; в пам'ять
Exit:
```

Стек

Стек – це спеціальна область пам'яті. Адресацією в цій області управляє регістр SP або вказівник стека. Використовується ця пам'ять в основному для тимчасового зберігання вмісту регістрів. Саме тимчасового. Найголовніше зрозуміти, як працює стек. Він функціонує за принципом перший прийшов останній пішов - LIFO («Last In - First Out» або «останнім прийшов - першим пішов»). Розглянемо цей процес на прикладі: хтось ставить книгу на стіл, потім зверху ще одну книгу. Перша книга лежить внизу. Що б витягнути її спочатку потрібно зняти верхню і тільки після цього буде доступ до першої.

Зазвичай стек використовується для збереження адрес повернення і передачі аргументів під час виклику процедур, також у ньому виділяється пам'ять для локальних змінних. Крім того, в стеку можна тимчасово зберігати значення регістрів.

Стек є невід'ємною частиною архітектури процесора і підтримується на апаратному рівні: у процесорі є спеціальні регістри (SS, BP, SP) і команди для роботи зі стеком. Схема організації стека в процесорі 8086 показана на рисунку 12.1.

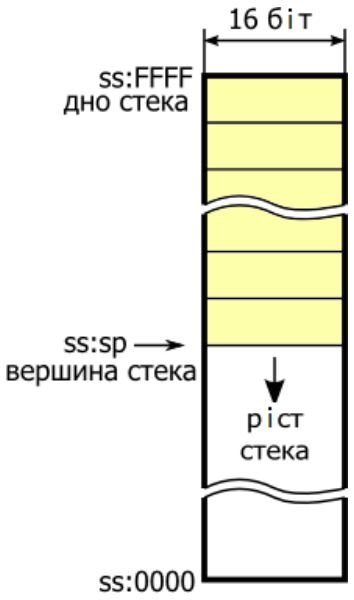


Рисунок 12.1 – Схема організації стека в процесорі 8086

Стек розміщується в оперативній пам'яті в сегменті стека, і тому адресується відносно початку сегментного регістра SS. Шириною стека називається розмір елементів, які можна поміщати в нього або витягувати. У нашому випадку ширина стека дорівнює двом байтам або 16 бітам. Регістр SP (вказівник стека) містить

адресу останнього доданого елемента. Ця адреса також називається вершиною стека. Протилежний кінець стека називається дном.

Дно стека знаходиться у верхніх адресах пам'яті. При додаванні нових елементів у стекзначення регістра SP зменшується, тобто стек росте в бік молодших адрес.

Для стека існують всього дві основні операції:

- додавання елемента на вершину стека (PUSH);
- витягування елемента з вершини стека (POP);

Додавання елемента в стек командою PUSH

Виконується командою PUSH. У цієї команди один операнд, який може бути безпосереднім значенням, 16-бітним регістром (у тому числі сегмент) або 16-бітної змінної в пам'яті. Команда працює наступним чином:

1. Значення в регістрі SP зменшується на 2 (так як ширина стека - 16 біт або 2 байти);
2. Операнд поміщається в пам'ять за адресою в SP.

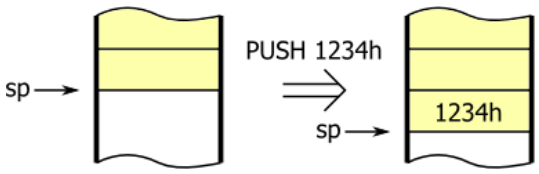


Рисунок 12.2 – Додавання елементу в стек

Приклади:

```
push -5    ;Помістити -5 в стек
push ax    ;Помістити AX в стек
push ds    ;Помістити DS в стек
push x     ;Помістити x в стек (змінна x визначена як слово)
```

Існують ще 2 команди для додавання в стек. Команда PUSHF поміщає в стек вміст регістра прапорів. Команда PUSHA поміщає в стек вміст всіх регістрів загального призначення в наступному порядку: AX, CX, DX, BX, SP, BP, SI, DI (значення DI буде на вершині стека). Значення SP поміщається те, яке було до виконання команди. Обидві ці команди не мають операндів.

Витягування елемента з стека командою POP

Виконується командою POP. У цієї команди також є один операнд, який може бути 16- бітним регістром (у тому числі сегментним, але крім CS) або 16-бітною змінною в пам'яті. Команда працює наступним чином:

1. Операнд читається з пам'яті за адресою в SP;
2. Значення в регістрі SP збільшується на 2.

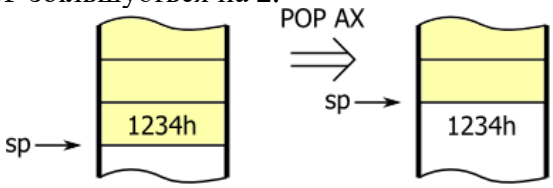


Рисунок 12.3 – Витягування елементів зі стеку

Зверніть увагу, що витягнутий зі стеку елемент не обнуляється і не затирається в пам'яті, а просто залишається як сміття. Він буде перезаписаний при поміщенні новогозначення в стек.

Приклади:

```
pop cx    ;Помістити значення з стека в CX
pop es    ;Помістити значення з стека в ES
pop x     ;Помістити значення з стека в змінну x
```

Відповідно, є ще 2 команди. POPF відновлює значення з вершини стека в регістр прапорів. POPA відновлює з стека всі регістри загального призначення (але при цьому значення для SP ігнорується).

Процедури

Процедурою називається код, який може виконуватися багаторазово і до якого можна звертатися з різних частин програми. Зазвичай процедури призначені для виконання якихось окремих, закінчених дій програми і тому їх іноді називають підпрограмами. В інших мовах програмування процедури можуть називатися функціями або методами, але по суті це все одне й те ж.

Команди CALL і RET

Для роботи з процедурами призначені команди CALL і RET. Крім того для позначення початку і закінчення процедури використовуються директиви PROC і ENDP, які потрібно вставляти після назви процедури відповідно перед тілом процедури та після неї. За допомогою команди CALL виконується виклик процедури. Ця команда працює майже так само, як команда безумовного переходу (JMP), але з однією відмінністю - одночасно в стек зберігається поточне значення регістра IP. Це дозволяє потім повернутися до того місця в коді, звідки була викликана процедура. В якості операнда вказується адреса переходу, яка може бути безпосереднім значенням (міткою), 16-розрядним регістром (крім сегментних) або коміркою пам'яті, яка містить адресу.

Повернення з процедури виконується командою RET. Ця команда відновлює значення з вершини стека в регістр IP. Таким чином, виконання програми триває з команди, яка є наступною відразу після команди CALL. Зазвичай код процедури закінчується цією командою. Команди CALL і RET не змінюють значення прапорів (крім деяких особливих випадків у захищеному режимі). Невеликий приклад різних способів виклику процедури:

```
mov ax, myproc
mov bx, myproc_addr
call myproc ; Виклик процедури (адреса перехода – мітка myproc)
```

```
call ax ; Виклик процедури по адресі в AX
call [myproc_addr] ; Виклик процедури по адресі в змінній
mov ax,4C00h
int 21h ; Завершення програми
;-------------------------------------------------------------
-------
;Процедура, яка додає між собою 3 числа
myproc PROC
    add ax,bx
    add ax,cx
    ret ; Вихід з процедури
myproc ENDP
;-------------------------------------------------------------
-------
myproc_addr dw myproc ;Змінна с адресом процедури
```

Близький та далекий виклик процедур

Існує два типи викликів процедур. ***Близьким*** називається виклик процедури, яка знаходиться в поточному сегменті коду. ***Далекий*** виклик - це виклик процедури в іншому сегменті. Відповідно існують 2 види команди RET - для ближнього і далекого повернення. Компілятор автоматично визначає потрібний тип машинної команди, тому в більшості випадків не потрібно про це турбуватися.

Передача параметрів

Дуже часто виникає необхідність передати процедурі якісь параметри. Наприклад, якщо ви пишете процедуру для обчислення суми елементів масиву, зручно в якості параметрів передавати їй адреси масиву і його розмір. У такому випадку одну і ту ж процедуру можна буде використовувати для різних масивів у вашій програмі. Найпростіший спосіб передати параметри - це помістити їх в регістри перед викликом процедури.

Значення, що повертається

Крім передачі параметрів часто потрібно отримати якесь значення з процедури. Наприклад, якщо процедура щось обчислює, хотілося б отримати результат обчислення. А якщо процедура щось робить, то корисно дізнатися, завершилася дія успішно чи виникла помилка. Існують різні способи повернення значення з процедури, але найбільш популярний – це помістити значення в один з регістрів. Зазвичай для цієї мети використовують регістри AL і AX.

Збереження регістрів

Хорошим прийомом є збереження регістрів, які процедура змінює в ході свого виконання. Це дозволяє викликати процедуру з будь-якої частини коду і не турбуватися, що значення в регістрах будуть зіпсовані. Зазвичай регістри зберігаються в стеку за допомогою команди PUSH, а перед поверненням з процедури відновлюються командою POP. Звичайно, що відновлювати їх треба в зворотному порядку.

```
Myproc PROC
```

```
push bx   ; Збереження регістрів push cx
push si
...  ; Код процедури
pop si    ; Відновлення регістрів pop cx
pop bx
ret  ; Вихід з процедури myproc ENDP
```

2. Хід роботи

1. Вивчити основні поняття та оператори, які використовуються в програмі, поданій в теоретичних відомостях.

2. Ввімкнути ЕОМ.

3. Написати програму яка просить користувача ввести змінну **X**, коефіцієнти **A**, **B**. Потім програма обчислює формулу відповідно до варіанту і видає результат обчислення на екран. Програма повинна працювати як з від'ємними так і з додатними цілими числами.Після цього програма запитує користувача чи він хоче завершити програму і пропонує ввести символи 'Y' ('y') або 'N' ('n'). Якщо користувач вводить символ 'Y' ('y') – програма завершується, а якщо користувач вводить символ N' ('n'), то програма переходить на початок, тобто знову пропонує ввести змінну і коефіцієнти, після чого знову видає результат обчислення. В коді програми повинні використовуватись підпрограми і стек.

№ варіанту	Завдання
1	$Y=\begin{cases}\frac{(X+3A)^2}{2}-1B & x<0\\ \frac{(X-3A)^2}{2}-1B & x\geq 0\end{cases}$
2	$Y=\begin{cases}\frac{(2X-5)}{4A}-3B & x<10\\ \frac{(2X-5)}{4A}+3B & x\geq 10\end{cases}$
3	$Y=\begin{cases}\frac{(3X+4A)^2-5}{2B} & x<20\\ \frac{(3X-4A)^2-5}{2B} & x\geq 20\end{cases}$
4	$Y=\begin{cases}\frac{X^3-2A}{3}-4B & x<30\\ \frac{X^3-2A}{3}+4B & x\geq 30\end{cases}$

5	$Y=\begin{cases}\dfrac{(X\cdot A+2)^2}{10B}-3 & x<40\\ \dfrac{(X\cdot A-2)^2}{10B}-3 & x\geq 40\end{cases}$
6	$Y=\begin{cases}\dfrac{2X-4A}{5}+2B & x<6\\ \dfrac{2X-4A}{5}-2B & x\geq 6\end{cases}$
7	$Y=\begin{cases}\left(\dfrac{2A-X}{5}+3B\right)^2 & x<7\\ \left(\dfrac{2A-X}{5}-3B\right)^2 & x\geq 7\end{cases}$
8	$Y=\begin{cases}\dfrac{2XA-4}{3B}+4 & x<8\\ \dfrac{2XA-4}{3B}-4 & x\geq 8\end{cases}$
9	$Y=\begin{cases}\left(\dfrac{4A-X}{5B}\right)^3+1 & x<9\\ \left(\dfrac{4A+X}{5B}\right)^3+1 & x\geq 9\end{cases}$
10	$Y=\begin{cases}\dfrac{125A}{X\cdot B+4}+1 & x<100\\ \dfrac{125A}{X\cdot B-4}+1 & x\geq 100\end{cases}$
11	$Y=\begin{cases}\dfrac{101}{5\cdot BX}+4A & x<12\\ \dfrac{101}{5\cdot B-X}-4A & x\geq 12\end{cases}$
12	$Y=\begin{cases}\dfrac{95A}{3\cdot B+X^2}-4 & x<25\\ \dfrac{95A}{3\cdot B-X^2}-4 & x\geq 25\end{cases}$
13	$Y=\begin{cases}\dfrac{88A}{(4\cdot B-X)^2}+8 & x<15\\ \dfrac{88A}{(4\cdot B+X)^2}+8 & x\geq 15\end{cases}$
14	$Y=\begin{cases}\dfrac{200A}{X^3+1}-4B & x<31\\ \dfrac{200A}{X^3+1}+4B & x\geq 31\end{cases}$

15	$Y=\begin{cases}\dfrac{115B}{X^2-3A}+1 & x<42\\ \dfrac{115B}{X^2-3A}-1 & x\geq 42\end{cases}$
16	$Y=\begin{cases}\dfrac{62}{3A-X^3}+2B & x<55\\ \dfrac{62}{3A+X^3}+2B & x\geq 55\end{cases}$
17	$Y=\begin{cases}\dfrac{50}{5A-X^4}+8B & x<99\\ \dfrac{50}{5A-X^4}-8B & x\geq 99\end{cases}$
18	$Y=\begin{cases}\left(\dfrac{2A-X}{5}+3B\right)^2 & x<7\\ \left(\dfrac{2A-X}{5}-3B\right)^2 & x\geq 7\end{cases}$

4. Скопмілювати програму.
5. Перевірити програму на наявність помилок, у випадку їх наявності — виправити іперейти до кроку 3.
6. Скомпонувати програму.
7. Якщо в результаті компонування отриманий виконуваний файл, то виконати програму.
8. Результатом виконання лабораторної роботи повинна бути програма, яка виводить вкомандній стрічці результат обчислення формули, яка задана в п. 3.

3. **Зміст звіту**

1. Титульна сторінка, оформлена відповідно до зразка.
2. Тема роботи
2. Мета роботи.
3. Завдання до виконання лабораторної роботи.
7. Текст (лістинг) програми.
8. Скріншот екрану з результатами асемблювання, компонування та виконання програми.
9. Висновки.

4. **Контрольні запитання**

1. Яка команда використовується для виклику процедури в мові програмування Асемблер?
2. Яка команда використовується для завершення процедури в мові програмуванняАсемблер?
3. Як оголошується змінна в мові програмування Асемблер?
4. Які є типи змінних в мові програмування Асемблер?

5. Що таке стек?
6. Які команди використовуються для роботи з стеком?
7. Що таке процедура?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Білак, Ю.Ю., Данько-Товтин, Л.Я. Системи числення: методичні рекомендації з базової теми дисципліни «Інформатика». Ужгород: ДВНЗ «УжНУ», 2015. 24 с.
2. Батрак, Ю.А. Вступ до програмування. [Електронний ресурс]. Доступно на: https://lib.chmnu.edu.ua/pdf/posibnuku/230/4.pdf . Дата доступу: 01.09.2022.
3. Матвієнко, М.П., Розен, В.П., Закладний, О.М. Архітектура комп’ютера: навчальний посібник. Київ: Видавництво Ліра – К, 2016. 264 с.
4. Крупельницький, Л.В., Снігур, А.В., Богомолов, С.В. Архітектура комп’ютерів. Частина 1: Лабораторний практикум. Вінниця: ВНТУ, 2020. 124 с.
5. Антоненко, О.В., Бардус, І.О. Архітектура комп’ютера та конфігурування комп’ютерних систем (на основі фундаменталізованого підходу): навчальний посібник. Бердянськ, 292 с.
6. Тарарака, В.Д. Архітектура комп'ютерних систем: навчальний посібник. Житомир: ЖДТУ, 2018. 383 с.
7. Free System Information Tools. [Електронний ресурс]. Доступно на: https://www.lifewire.com/free-system-information-tools-2625772 . Дата доступу: 01.09.2022.
8. Elmir Configurator. [Електронний ресурс]. Доступно на: https://elmir.ua/configurator/ . Дата доступу: [вказати дату доступу].
9. Макаренко, В.В. Програмні засоби проектування: Навчальний посібник. Київ: КПІ ім. Ігоря Сікорського, 2018. 244 с.
10. Литвинов, А.Л. Практикум із лабораторних робіт з архітектури комп’ютерних систем: навч. посібник. Харків: ХНУМГ ім. О. М. Бекетова, 2020. 48 с.

ЗМІСТ

ВСТУП 4
РОБОТА №1. СИСТЕМИ ЧИСЛЕННЯ 5
РОБОТА №2. ЦЕНТРАЛЬНІ І ЗОВНІШНІ ПРИСТРОЇ ПК 12
РОБОТА №3. ВСТУП ДО СИСТЕМИ МОДЕЛЮВАННЯ ЕЛЕКТРОННИХ СХЕМ MULTISIM 20
РОБОТА №4. ДОСЛІДЖЕННЯ ФУНКЦІОНУВАННЯ БЛОКА ЖИВЛЕННЯ СТАЦІОНАРНИХ КОМП'ЮТЕРІВ 25
РОБОТА №5. ДОСЛІДЖЕННЯ ЛОГІЧНИХ СХЕМ І СХЕМ ПАМ'ЯТІ НА ДІОДАХ І ТРАНЗИСТОРАХ 31
РОБОТА №6. ДОСЛІДЖЕННЯ ТРИГЕРІВ КОМП'ЮТЕРНИХ СИСТЕМ 37
РОБОТА №7. ПОБУДОВА І ДОСЛІДЖЕННЯ ЛІЧИЛЬНИКІВ 48
РОБОТА №8. ДОСЛІДЖЕННЯ АРИФМЕТИКО-ЛОГІЧНОГО ПРИСТРОЮ 53
РОБОТА №9. НАПИСАННЯ ПРОСТОЇ ПРОГРАМИ НА МОВІ ASSEMBLER 58
РОБОТА №10. КОМАНДИ ПОРІВНЯННЯ НА АСЕМБЛЕРІ 69
РОБОТА №11. АРИФМЕТИЧНІ ОПЕРАЦІЇ В МОВІ ПРОГРАМУВАННЯ ASSEMBLER 78
РОБОТА №12. ВИКОРИСТАННЯ ЗМІННИХ, СТЕКУ ТА ПІДПРОГРАМ В МОВІ ПРОГРАМУВАННЯ АСЕМБЛЕР 88
СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ 97
ЗМІСТ 98

Навчальне видання

Укладачі:
ВОДКА Олексій Олександрович
ЛИТВИНОВ Анатолій Леонідович
МАРУСЕНКО Олексій Миколайович
ТРУБАЄВ Олександр Іванович
ПІД'ЯЧИЙ Гліб Юрійович

ЛАБОРАТОРНИЙ ПРАКТИКУМ З АРХІТЕКТУРИ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

Методичні вказівки для виконання лабораторних робіт
з курсу «Архітектура обчислювальних систем»
для студентів спеціальностей 122 «Комп'ютерні науки»,
113 «Прикладна математика» усіх форм навчання вищих навчальних закладів

Відповідальний за випуск проф. Львов Г.І.
Роботу до друку рекомендував доц. Федоров В.О.

В авторській редакції

План 2023 р., Поз.150
Підпісано до друку 1.12.2023р.
Гарнітура Times New Roman.
Обсяг – 4,7