МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ

Кафедра кібербезпеки та DATA-технологій

Факультет №6

МЕТОДИЧНІ МАТЕРІАЛИ

ДО СЕМІНАРСЬКИХ ЗАНЯТЬ

з навчальної дисципліни
«Архітектура та структурно-логічні основи ЕОМ»

вибіркових компонент
освітньої програми *першого*(*бакалаврського*) рівня вищої
освіти125 «Кібербезпека та захист інформації»
(«Безпека інформаційних та комунікаційних систем»)

м. Харків

2023 р.

ЗАТВЕРДЖЕНО
Науково-методичною радою Харківського національного університету внутрішніх справ
Протокол від 30.08.2023 № 7

СХВАЛЕНО
Вченою радою факультету №6 ХНУВС Протокол від 25.08.2023 № 7

ПОГОДЖЕНО
Секцією науково-методичної ради ХНУВС з технічних дисциплін
Протокол від 29.08.2023 № 7

Розглянуто на засіданні кафедри кібербезпеки та DATA-технологій (*протокол від 15.08.2023 № 8)*

Розробники:
Доцент кафедри кібербезпеки та DATA-технологій ХНУВС, кандидат технічних наук, доцент Юрій ГОРЕЛОВ

Рецензенти:
Доцент кафедри програмного забезпечення ХНУРЕ, к.т.н., доцент, Олексій Лановий;
Професор кафедри протидії кіберзлочинності факультету № 4 ХНУВС, к.т.н., доцент, Віталій Носов

Розподіл часу навчальної дисципліни за темами (денна форма навчання)

Номер та найменування теми	Кількість годин відведених на вивчення навчальної дисципліни						Література, сторінки	Вид контролю
	Всього	з них:						
		Лекції	Семінарські заняття	Практичні заняття	Лабораторні заняття	Самостійна робота		
Тема № 1. Логіко-математичні основи інформатики	24	10	–	10	–	8	[1]	
Тема № 2. Архітектура обчислювальних систем	22	6	4		-	10	[1,3,8,9, 11]	
Тема № 3. Компютерні мережі	20	4	4	-	-	10	[1,3,7,8, 9]	
Тема № 4. Програмне забезпечення	24	4	4	2	-	14	[1,3,7,8, 9]	
Всього за семестр №1	**90**	**24**	**12**	**12**	**-**	**42**		**екзамен**

Розподіл часу навчальної дисципліни темами (заочна та дистанційна форма навчання)

Номер та найменування теми	Кількість годин відведених на вивчення навчальної дисципліни						Література, сторінки	Вид контролю
	Всього	з них:						
		Лекції	Семінарські заняття	Практичні заняття	Лабораторні заняття	Самостійна робота		
Тема № 1. Логіко-математичні основи інформатики	20	2	4	–	–	20	[1]	
Тема № 2. Архітектура обчислювальних систем	26	2	–	2	-	20	[1,3,7, 9]	
Тема № 3. Компютерні мережі	24	–	–	-	-	20	[1,3,7, 8,9]	
Тема № 4. Програмне забезпечення	20	4	4	2	-	20	[1,3,7,	
Всього за семестр №1	**90**	**2**	–	8	**-**	**80**		**Екзамен**

3.Методичні вказівки до семінарських занять

Тема №2 Архітектура обчислювальних систем

Семінарське заняття: Подання даних в ЕОМ.

Мета заняття: Вивчити форми подання даних в ЕОМ

Тривалість: 2 год

Навчальні питання

1. Подання чисел з фіксованою точкою (комою);
2. Подання чисел з плаваючою точкою (комою);
3. Прямий та зворотній коди
4. Подання символьних даних.

Подання чисел з фіксованою точкою (комою);

При поданні числа X у **формі з фіксованою точкою** вказуються знак числа (sign X) і модуль числа (mod X) в q-ичному коді. Іноді така форма подання чисел називається природною формою. Місце точки (коми) стале для всіх чисел і в процесі розв'язування задач не змінюється. Знак додатного числа кодується цифрою «0», а знак від'ємного числа - цифрою «1».

У загальному випадку розрядна сітка ЕОМ для розміщення чисел у формі з фіксованою точкою показана на рисунку 1.2. Тут показано n розрядів для подання цілої частини числа і r розрядів - для дробової частини числа.

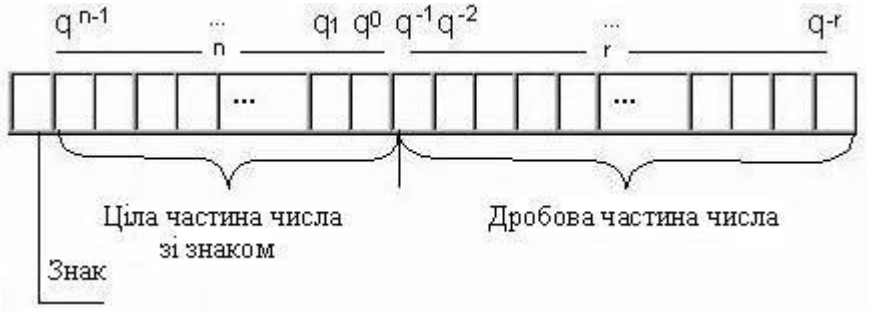

Рис. 1.2.

При заданих n і r діапазон зміни модулів чисел, коди яких можуть бути подані в даній розрядній сітці, визначається нерівністю:

$$q^{-r} \le |X| \le q^n - q^{-r}.$$

Використання форми з фіксованою точкою для подання змішаних (з цілою і дробовою частиною) чисел в ЕОМ практично не зустрічається. Як правило, використовуються в ЕОМ числа або з дробовою арифметикою (n=0), або з цілочисловою арифметикою (r=0).

Подання чисел з плаваючою точкою (комою

Форма подання чисел із фіксованою точкою спрощує апаратну реалізацію ЕОМ, зменшує час виконання машинних операцій, проте при розв'язуванні задач на ЕОМ необхідно постійно стежити за тим, аби всі початкові дані, проміжні і остаточні результати знаходилися в допустимому діапазоні подання. Якщо цього не дотримуватися, то можливе переповнювання розрядної сітки, і результат обчислень буде невірним. Від цих недоліків значною мірою вільні ЕОМ, що використовують **форму подання чисел із плаваючою точкою, або нормальну форму**.

У нормальній формі число подається у вигляді добутку $X=mq^p$, де m - мантиса числа; q - основа системи числення; p – порядок.

Для задання числа в нормальній формі потрібно задати знаки **мантиси** і **порядку**, їх модулі в q-ичному коді, а також основу системи числення. Нормальна форма подання чисел неоднозначна, бо взаємна зміна m і p призводить до плавання точки (коми). Звідси пішла назва форми подання чисел.

Для однозначності подання чисел в ЕОМ використовується **нормальна нормалізована форма**, в якій положення точки завжди задається перед значущою цифрою мантиси, тобто виконується умова:

$$q^{-1} \le |m| \le q^0 = 1$$

У загальному випадку розрядну сітку ЕОМ для розміщення чисел у нормальній формі можна подати у вигляді, зображеному на рис. 1.3.

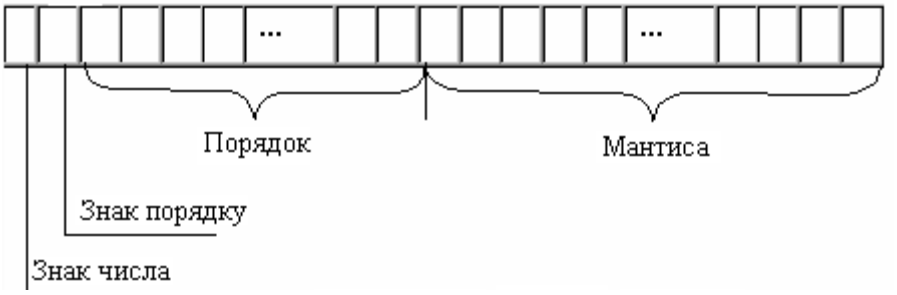

Рис. 1.3.

Розрядна сітка містить:
• розряд для знаку мантиси (числа);
• *r* цифрових розрядів для *q*-ичного коду модуля мантиси;
• розряд для коду знаку порядку;
• *s* розрядів для *q*-ичного коду модуля порядку.

Діапазон подання модулів чисел у нормальній нормалізованій формі визначається такою нерівністю:

$$q^{-1}q^{-(q^s-1)} \le |X| \le (1-q^{-r})^{(q^s-1)}$$

У конкретній ЕОМ діапазон подання чисел із плаваючою точкою залежить від основи системи числення і числа розрядів для подання порядку.

При цьому в однакових за довжиною форматах чисел із плаваючою точкою зі збільшенням основи системи числення істотно розширюється діапазон експонованих чисел.

Точність обчислень при використанні формату з плаваючою точкою визначається числом розрядів мантиси *r*. Вона збільшується зі збільшенням числа розрядів.

При **поданні інформації у вигляді десяткових багаторозрядних чисел** кожна десяткова цифра замінюється двійково-десятковим кодом. Для прискорення обміну інформацією, економії пам'яті і зручності операцій над десятковими числами передбачаються спеціальні формати їх подання: зонний (розпакований) і **упакований. Зонний формат** використовується в операціях введення. Для цього в ЕОМ є спеціальні команди упаковки і розпаковування десяткових чисел.

3. Прямий та зворотній коди

Для зберігання чисел і виконання різних операцій над ними їх подають різними кодами: **прямим, зворотним і додатковим**. Для подання чисел зі знаками в пам'яті ЕОМ використовується прямий код. Для позначення прямого коду числа X використовується запис виду $[X]$.

Правило подання *q*-ичного коду числа в **прямому коді** має вигляд:

$$\left[X\right]_{np} = \begin{cases} 0x_{n-1}x_{n-2}\ldots x_1x_0, x_{-1}\ldots x_{-m}, & якщо\ X \ge 0; \\ 1x_{n-1}x_{n-2}\cdots x_1x_0, x_{-1}\cdots x_{-m}, & якщо\ X < 0; \end{cases}$$

де x_i- значення цифри в *i*-му розряді початкового коду.

Тут старший біт несе інформацію про знак числа. Якщо він приймає значення 0, то знак числа «+»; якщо значення 1 - то знак числа «-».

Наприклад, для двійкового коду

X_2 = +11001011 $[X_2]$ = 0.11001011;
X_2 = -01101011 $[X_2]$ = 1.01101011.

При поданні чисел у прямому коді реалізація арифметичних операцій в ЕОМ повинна передбачати різні дії з модулями чисел залежно від їх знаків. Так, додавання в прямому коді чисел з однаковими знаками виконується досить просто. Числа складаються і сумі присвоюється код знака доданків. Значно складнішою є операція алгебраїчного додавання в прямому коді чисел із різними знаками. У цьому випадку доводиться визначати більше за модулем число, виконувати віднімання чисел і привласнювати різниці знак більшого за модулем числа. Для спрощення виконання операцій алгебраїчного додавання в ЕОМ використовуються спеціальні коди, які дозволяють звести цю операцію до операції арифметичного додавання. В ЕОМ застосовуються спеціальні зворотний і додатковий коди. Вони утворюються з прямих кодів чисел, причому спеціальний код додатного числа дорівнює його прямому коду.

Для позначення **зворотного коду** числа X_q використовується запис виду $[X_q]_{звор}$.

Правило подання *q*-ичного коду числа у зворотному коді має вигляд:

$$[X_q]_{звор} = \begin{cases} 0x_{n-1}x_{n-2}\cdots x_1x_0, x_{-1}\cdots x_{-m}, & якщо\ X \ge 0; \\ 1\bar{x}_{n-1}\bar{x}_{n-2}\cdots \bar{x}_1\bar{x}_0, \bar{x}_{-1}\cdots \bar{x}_{-m}, & якщо\ X < 0; \end{cases}$$

Тут інверсія цифри x_i визначається зі співвідношення:

$$\bar{x}_j = (q-1) - x_j, \quad -m \le j \le n-1,$$

де q - основа системи числення; x_j значення цифри в j-ому розряді початкового коду.

Для двійкової системи числення, якщо x_j=1, то $\bar{x}_j$=0 і навпаки. Звідси можна сформулювати окреме правило одержання зворотного коду для від'ємних двійкових чисел. Для перетворення прямого коду двійкового від'ємного числа у зворотний код і навпаки необхідно знаковий розряд залишити без зміни, а в решті розрядах нулі замінити на одиниці, а одиниці на нулі.

Наприклад:

x_2=+11011001, $[X_2]_{пр}$ =0.11011001, $[X_2]_{звор}$=0.11011001.

x_2=- 01011101, $[X_2]_{пр}$ =1.01011101, $[X_2]_{звор}$=1.10100010.

Для позначення **додаткового коду** числа X_q використовується запис виду $[X_q]_{дод}$. Правило подання q-ичного коду числа в додатковому коді має вигляд:

$$[X_q]_{дод} = \begin{cases} 0x_{n-1}x_{n-2}\cdots x_1x_0, x_{-1}\cdots x_{-m}, & якщо\ X \ge 0; \\ 1\bar{x}_{n-1}\bar{x}_{n-2}\cdots \bar{x}_1\bar{x}_0, \bar{x}_{-1}\cdots \bar{x}_{-m} + q^{-m}, & якщо\ X < 0; \end{cases}$$

Таким чином, для перетворення прямого коду q-ичного від'ємного числа в додатковий необхідно перетворити його у зворотний код і в молодший розряд додати одиницю.

Наприклад, для двійкових чисел:

x_2=+11011001, $[X_2]_{пр}$=0.11011001, $[X_2]_{дод}$=0.11011001.

x_2=-01011101, $[X_2]_{пр}$=1.01011101, $[X_2]_{дод}$=1.10100011.

При виконанні операції додавання чисел, поданих спеціальними q-ичними кодами знакові розряди беруть участь в операції поряд із цифровими розрядами. При цьому цифрові розряди доданків складаються як модулі чисел за правилами q-ичної арифметики. Знакові розряди і цифри перенесення зі старшого цифрового розряду при будь-якій основі системи числення складаються як однорозрядні двійкові коди. Якщо при цьому формується перенесення зі знакового розряду, то він має вагу одиниці молодшого розряду q_m при використанні зворотного коду і повинен бути доданий у молодший розряд результату. При використанні додаткового коду одиниця перенесення зі знакового розряду не береться до уваги, тобто відкидається.

Наприклад:

Десяткова запис	Двійкові коди	
10	0 0001010	
+ -3	+ 1 1111100	Зворотний код числа -3
7	0 0000110	
	→ +1	
	0 0000111	

Десяткова запис	Двійкові коди	
10	0 0001010	
+ -3	+ 1 1111101	Додатковий код числа -3
7	0 0000111	
	→ Перенесення відкидається	

При виконанні операції алгебраїчного додавання перед перетворенням прямих кодів доданків у спеціальні необхідно їх вирівняти за числом розрядів, якщо число розрядів доданків різне. Крім того, в деяких випадках може статися переповнення розрядної сітки. Ознакою переповнення розрядної сітки є така комбінація цифр у знакових розрядах доданків і результату: 0+0=1 або 1+1=0.

Результат складання спеціальних кодів чисел при переповненні розрядної сітки є невірним.

Подання символьних даних.

Існує множина систем подання даних. Однією з них, прийнятою в інформатиці та обчислювальній техніці, є ***двійкове кодування***. Вона грунтується на поданні даних послідовністю двох знаків: 0 і 1. Ці знаки називаються *двійковими цифрами* (*біт*, *binary digit*, *bit*).

Одним бітом подаються два поняття: 0 або 1 (*так* або *ні*, *чорне* або *біле*, *істина* або *неправда* і т. п.). Якщо кількість бітів збільшити до двох, то можна подати чотири різні поняття:

00 01 10 11.

Трьома бітами можна закодувати вісім різних понять:

000 001 010 011 100 101 110 111.

Зі збільшенням на одиницю кількості біт у системі двійкового кодування збільшується в два рази кількість значень, яку можна виразити в даній системі, тобто загальна формула має вигляд:

$$N=2^n,$$

де N - кількість незалежних закодованих значень;

n - розрядність двійкового кодування, прийнята в даній системі.

Якщо кожному символу абетки поставити у відповідність певне ціле число (наприклад порядковий номер), то за допомогою двійкового коду можна кодувати і текстову інформацію. Восьми двійкових розрядів достатньо для кодування 256 різних символів. Цього вистачить, аби подати різними комбінаціями восьми біт усі символи англійської та української мов, як рядкові, так і прописні, а також розділові знаки, символи основних арифметичних дій і деякі загальноприйняті спеціальні символи, наприклад символ "§".

2. Система *ASCII*

Інститут стандартизації США (*ANSI - American National Standard Institute*) ввів у дію систему кодування *ASCII* (*American Standard Code for Information Interchange* - стандартний код інформаційного обміну США). У системі *ASCII* закріплені дві таблиці кодування - *базова* і *розширена*. Базова таблиця закріплює значення кодів від 0 до 127, а розширена відноситься до символів із номерами від 128 до 255.

Перші 32 коди базової таблиці, починаючи з нульового, віддані виробникам апаратних засобів (у першу чергу виробникам комп'ютерів і друкуючих пристроїв). У цій області розміщуються управляючі коди, яким не відповідають ніякі символи мов, і, відповідно, ці коди не виводяться ні на екран, ні на пристрої друкування, але ними можна управляти виведенням інших даних.

Таблиця 1.1. Базова таблиця кодування ASCII

32	48	64 @	80 P	96 '	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 "	50 2	66 B	82 R	98 b	114 r
35 #	51 3	67 C	83 S	99 c	115 s
36 $	52 4	68 D	84 T	100 d	116 t
37 %	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 '	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75K	91 [	107 k	123 {
44 ,	60 <	76 L	92 \	108 l	124 |
45 -	61 =	77 M	93]	109 m	125 }
46 .	62 >	78 N	94 ^	110 n	126 ~
47 /	63 ?	79 O	95 _	111 o	127

Починаючи з коду 32 по код 127 розміщені коди символів англійського алфавіту, розділових знаків, цифр, символів арифметичних дій і деяких допоміжних символів. Базова таблиця кодування *ASCII* приведена в таблиці 1.1.

Аналогічні системи кодування текстових даних були розроблені в інших країнах. Проте підтримка виробників устаткування і програм вивела американський код *ASCII* на рівень міжнародного стандарту, і національним системам кодування прийшлося "відступити" у другу, розширену частину системи кодування, що визначається значеннями кодів від 128 до 255. Відсутність єдиного стандарту в цій області призвів до множинності одночасно чинних систем кодування.

Так, наприклад, кодування символів російської і української мов, відоме як кодування Windows-1251, було введено "ззовні" - компанією Microsoft, але, з огляду на широке поширення операційних систем та інших продуктів цієї компанії на терені СРСР, воно глибоко закріпилося і знайшло широке поширення. Це кодування використовується на більшості локальних комп'ютерів, які працюють на платформі Windows.

Якщо проаналізувати організаційні труднощі, пов'язані зі створенням єдиної системи кодування текстових даних, то можна прийти до висновку, що вони викликані обмеженим набором кодів (256). В той же час очевидно, що коли, наприклад, кодувати символи не вісьмирозрядними двійковими числами, а числами з більшою кількістю розрядів, то й діапазон можливих значень кодів стане набагато більшим. Система, яка грунтується на 16-розрядному кодуванні символів, одержала назву універсальної - *UNICODE*. Шістнадцять

розрядів дозволяють забезпечити унікальні коди для 65536 різних символів - цього достатньо для розміщення в одній таблиці символів більшості мов планети.

Незважаючи на тривіальну очевидність такого підходу, простий механічний перехід на дану систему довгий час стримувався із-за недостатніх ресурсів засобів обчислювальної техніки (у системі кодування UNICODE усі текстові документи автоматично стають удвічі довшими). В другій половині 90-х років технічні засоби досягли необхідного рівня ресурсів, і сьогодні спостерігається перекладання документів і програмних засобів на універсальну систему кодування. Для індивідуальних користувачів це додало більших турбот на узгодження документів, виконаних у різних системах кодування, але це труднощі перехідного періоду.

3. Стандарт Юнікод

Стандарт Юнікод (Unicode) запропонований в 1991 році некомерційною організацією «Консорціум Юнікоду» (Unicode Consortium, Unicode Inc.). Застосування цього стандарту дозволяє кодувати досить велике число символів із різних писемностей: у документах Unicode можуть бути сусідами китайські ієрогліфи, математичні символи, літери грецького алфавіту, латиниці і кирилиці, при цьому стає непотрібним перемикання кодових сторінок.

Універсальна система кодування (Юнікод) є набором графічних символів і способом їх кодування для комп'ютерної обробки текстових даних.

Графічні символи - це символи, які мають видиме зображення. Графічним символам протиставляються управляючі символи і символи форматування.

Графічні символи включають в себе такі групи:

- літери, що містяться принаймні в одному з обслуговуваних алфавітів;
- цифри;
- знаки пунктуації;
- спеціальні знаки (математичні, технічні, ідеограми тощо);
- роздільники.

Отже, Юнікод - це система для лінійного подання тексту. Символи, що мають додаткові над- або підрядкові елементи, можуть бути подані у вигляді побудованої за певними правилами послідовності кодів (складний варіант, composite character) або у вигляді єдиного символу (монолітний варіант, precomposed character). На даний момент вважається, що всі літери великих писемностей у Юнікод внесені, і якщо символ доступний у складному варіанті, дублювати його в монолітному вигляді не потрібно.

Графічні символи в Юнікоді діляться на протяжні і непротяжні (без ширини). Непротяжні символи при відображенні не займають місця в рядку. До них належать, зокрема, знаки наголосу та інші діакритичні знаки. Як протяжні, так і непротяжні символи мають власні коди. Протяжні символи називаються також базовими (base characters), а непротяжні - модифікованими (combining characters); причому останні не можуть зустрічатися самостійно. Наприклад, символ «á» може бути поданий як послідовність базового символу «a» (U+0061) і модифікуючого символу «’» (U+0301) або як монолітний символ «á» (U+00C1).

Особливий тип модифікованих символів - селектори варіантів накреслення (variation selectors). Вони діють тільки на ті символи, для яких такі варіанти визначені.

Для сумісності зі старими 16-бітними системами була винайдена система UTF-16, де перші 65 536 позицій, за винятком позицій з інтервалу U+D800 ... U+DFFF, відображаються безпосередньо як 16-бітові числа, а решта подаються як «сурогатні пари» (перший елемент пари з області U+D800 ... U+DBFF, другий елемент пари з області U+DC00 ... U+DFFF). Для сурогатних пар була використана частина кодового простору (2048 позицій), відведеного «для окремого використання».

Оскільки в UTF-16 можна відобразити тільки $2^{20}+2^{16}$-2048 (1 112 064) символів, то це число і було вибрано за остаточну величину кодового простору Юнікоду (діапазон кодів: 0x000000-0x10FFFF).

Кодовий простір Юнікоду розбито на 17 площин (planes) по 2^{16} (65 536) символів. Нульова площину (plane 0) називається базовою (basic) і містить символи найбільш вживаних писемностей. Решта площин - додаткові (supplementary). Перша площина (plane 1) використовується в основному для історичних писемностей, друга (plane 2) - для рідко використовуваних ієрогліфів китайського письма, третя (plane 3) зарезервована для архаїчних китайських ієрогліфів. Площини 15 і 16 виділені для окремого вживання.

Для позначення символів Unicode використовується запис виду «U+xxxx» (для кодів 0 ... FFFF), або «U+xxxxx» (для кодів 10000 ... FFFFF), або «U+xxxxxx» (для кодів 100000 ... 10FFFF), де xxx - шістнадцятиричні цифри. Наприклад, символ «я» (U+044F) має код 044F16 = 110310.

Теми доповідей

1. Подання даних в ЕОМ.
2. Числа з фіксованою точкою.
3. Числа з плаваючою точкою.
4. Кодування даних в інформатиці. Таблиця кодування ASCII.

Література

1. Зацеркляний М.М. Компютерні основи систем кібербезпеки: навч. посібник/ Зацеркляний М.М., Струков В.М.-Харків: Тов. «В деле», 2017.- 292 с.

2. Інформатика: Комп’ютерна техніка. Комп’ютерні технології: Підручник для студентів вищих навчальних закладів / За ред. Пушкаря О.І. – К.: Каравела, 2004.
3. Глинський Я. М. Практикум з інформатики: Навчальний посібник. – Львів: "Деол", 2001. – 224 с.

Тема 2 Архітектура обчислювальних систем

Семінарське заняття: Вузли обчислювальних машин

Мета заняття: Вивчити осносні вузли ЕОМ

Тривалість: 2 год

Навчальні питання

1. Тригери
2. Дешифратори;
3. Регістри;
4. Загальна шина

Тригери

Тригер – це найпростіший послідовних пристрій, який може тривалий час знаходитися в одному з декількох можливих стійких станів і переходити з одного стану в інший під впливом вхідних сигналів. Послідовними називаються такі логічні пристрої, вихідні сигнали яких визначаються не тільки сигналами на входах, а й передісторією їх роботи, тобто станом елементів пам'яті. Тригер - один із базових елементів цифрової техніки.

За способом роботи з сигналами розрізняють асинхронні, синхронні і змішані тригерні схеми, статичні та динамічні схеми.

Асинхронний тригер змінює свій стан безпосередньо в момент появи відповідного інформаційного сигналу. ***Синхронні тригери*** реагують на інформаційні сигнали тільки за наявності відповідного сигналу на так званому вході синхронізації С (clock). Цей вхід позначається термінами «стрибок», «такт». Синхронні тригери у свою чергу діляться на тригери зі статичним (статичні) і динамічним (динамічні) управлінням на вході синхронізації С. ***Статичні тригери*** сприймають інформаційні сигнали при подачі на вхід логічної одиниці (прямий вхід) або логічного нуля (інверсний вхід). ***Динамічні тригери*** сприймають інформаційні сигнали при зміні (перепаді) сигналу на вході С від 0 до 1 (прямий динамічний С-вхід) або від 1 до 0 (інверсний динамічний С-вхід).

Статичні тригери у свою чергу діляться на одноступеневі (однотактні) та двоступеневі (двотактні). В ***одноступеневому тригері*** є одна ступінь запам'ятовування інформації, а в ***двохступеневому*** - дві таких ступені. Спочатку інформація записується в першу ступінь, а потім переписується в другу і з'являється на виході. Двоступеневий тригер позначається ТТ.

За структурною побудовою розрізняють однотактні, двотактні тригери і тригери з динамічним управлінням.

За способом реакції на перешкоди є ***прозорі*** і ***непрозорі*** тригери. Непрозорі, в свою чергу, діляться на ***проникні*** і ***непроникні***. За функціональним призначенням розрізняють RS, D, JK, T, RR, SS, EE, DV тригери.

При виготовленні тригерів застосовуються переважно напівпровідникові прилади (як правило, польові транзистори), у минулому - електронні лампи. На сьогодні логічні схеми, в тому числі з використанням тригерів, створюються в інтегрованих середовищах розробки під різні програмовані логічні інтегральні схеми (ПЛІС). Використовуються в основному в обчислювальній техніці для організації компонентів обчислювальнихсистем: процесорів, регістрів, лічильників, оперативної пам’яті.

За функціональними можливостями тригери діляться на такі класи:

• з роздільною установкою станів 0 і 1 (RS-тригери). Якщо тригер є синхронним - додається вхід синхронізації С;

• універсальні (JK-тригери);

• з прийманням інформації одним входом D (D-тригери, або тригери затримки);

• з рахунковим входом Т (Т-тригери).

Кожен тип тригера має власну таблицю роботи (таблицю істинності). Вихідний стан тригера, як правило, позначається літерою Q. Індекс біля літери означає стан до подачі сигналу (t) або після подачі сигналу (t +1).

Якщо тригер синхронний, то існує також додатковий вхід синхронізації. Аби такий тригер врахував інформацію на синхронних входах, на вході синхронізації необхідно сформувати активний фронт (як правило, додатний фронт).

Регістри

Регістр – це послідовний логічний пристрій, що використовується для зберігання *n*-розрядних двійкових чисел і виконання перетворень над ними. Регістр є упорядкованою послідовністю тригерів, число яких відповідає числу розрядів у слові. З кожним регістром, як правило, пов'язаний комбінаційний цифровий пристрій, за допомогою якого забезпечується виконання деяких операцій над словами. Фактично будь-який цифровий пристрій можна подати у вигляді сукупності регістрів, з'єднаних один із одним за допомогою комбінаційних цифрових пристроїв.

Основою побудови регістрів є D-тригери.

Типовими є такі операції:

• приймання слова в регістр;
• передача слова з регістру;
• порозрядні логічні операції;
• зсув слова вліво або вправо на задане число розрядів;
• перетворення послідовного коду слова в паралельний і навпаки;
• установка регістра в початковий стан (скидання).

Регістри класифікуються за такими видами:

• накопичувальні (регістри пам'яті, зберігання);
• зсувні.

У свою чергу зсувні регістри діляться:

за способом введення-виведення інформації:

• паралельні - запис і зчитування інформації відбувається одночасно на всі входи і з усіх виходів;
• послідовні - запис і зчитування інформації відбувається в перший тригер, а та інформація, яка була в цьому тригері, перезаписується в наступний - те ж саме відбувається й з іншими тригерами;
• комбіновані;

за напрямом передачі інформації:

• односпрямовані;
• реверсні.

за основою системи числення:

• виконавчі;
• трійчасті;
• десяткові.

Дешифратори

Дешифраторами називаються комбінаційні пристрої, які перетворюють *n*-розрядний двійковий код у логічний сигнал, що з'являється на тому виході, десятковий номер якого відповідає бінарному коду.

Дешифратор працює за таким принципом: нехай дешифратор має N входів, на них подано двійкове слово $x_{N-1}x_{N-2}\ldots x_0$, тоді на виході матимемо такий код розрядності менший або рівний 2^N, що розряд, номер якого дорівнює вхідному слову, приймає значення одиниці, всі інші розряди дорівнюють нулю. Очевидно, що максимально можлива розрядність вихідного слова дорівнює 2^N. Такий дешифратор називається ***повним***. Якщо частина вхідних наборів не використовується, то число виходів менше за 2^N, і дешифратор є ***неповним***.

Часто дешифратори доповнюються входом дозволу роботи E. Якщо на цей вхід надходить одиниця, то дешифратор функціонує, в іншому випадку на виході дешифратора виробляється логічний нуль незалежно від вхідних сигналів.

Існують дешифратори з інверсними виходами, у такого дешифратора вибраний розряд є нулем. Функціонування дешифратора описується системою кон'юнкцій:

$$F_0 = \bar{\delta}_{N-1}\bar{\delta}_{N-2}\cdots\bar{\delta}_1\bar{\delta}_0 E$$
$$F_1 = \bar{\delta}_{N-1}\bar{\delta}_{N-2}\cdots\bar{\delta}_1 x_0 E$$
$$F_2 = \bar{\delta}_{N-1}\bar{\delta}_{N-2}\cdots\bar{\delta}_1\bar{\delta}_0 E$$
$$\ldots\ldots\ldots\ldots$$
$$F_{2N-2} = x_{N-1}x_{N-2}\cdots x_1\bar{\delta}_0 E$$
$$F_{2N-1} = x_{N-1}x_{N-2}\cdots x_1\delta_0 E$$

Зворотне перетворення здійснює шифратор.

Лічильник – це пристрій, на виходах якого дістається двійковий (двійково-десятковий) код, що визначається числом імпульсів, які надійшли. Лічильники можуть будуватися на Т-тригерах. Основний параметр лічильника - модуль рахунку - максимальне число одиничних сигналів, яке може бути перелічене лічильником. Лічильники позначають через СТ (counter).

Лічильники класифікують:

за модулем рахунку:

• двійково-десяткові;
• виконавчі;
• з довільним постійним модулем рахунку;
• зі змінним модулем рахунку;

за напрямом рахунку:

• сумуючі;
• віднімаючі;
• реверсні;

за способом формування внутрішніх зв'язків:
- з послідовним перенесенням;
- з паралельним перенесенням;
- з комбінованим перенесенням;
- кільцеві.

Загальна шина

Класична узагальнена структурна схема комп'ютера подається на рис. 2.3. При цьому використовуються такі позначення:

ПУ (пристрій управління) організовує автоматичне виконання програм і функціонування обчислювальної машини. Основне завдання ПУ - вироблення управляючих сигналів і розподіл їх ланцюгом управління.

АЛП (арифметико-логічний пристрій) призначений для виконання арифметичних і логічних операцій над даними, які надходять.

ОЗП (оперативний запам’ятовуючий пристрій) є масивом запам'ятовуючих елементів, організованих у вигляді запам'ятовуючих комірок і здатних зберігати деяку одиницю інформації.

Пристрій введення перетворює вхідну інформацію у послідовність електричних сигналів.

Пристрій виведення перетворює послідовність електричних сигналів у форму, зручну для сприйняття користувачем.

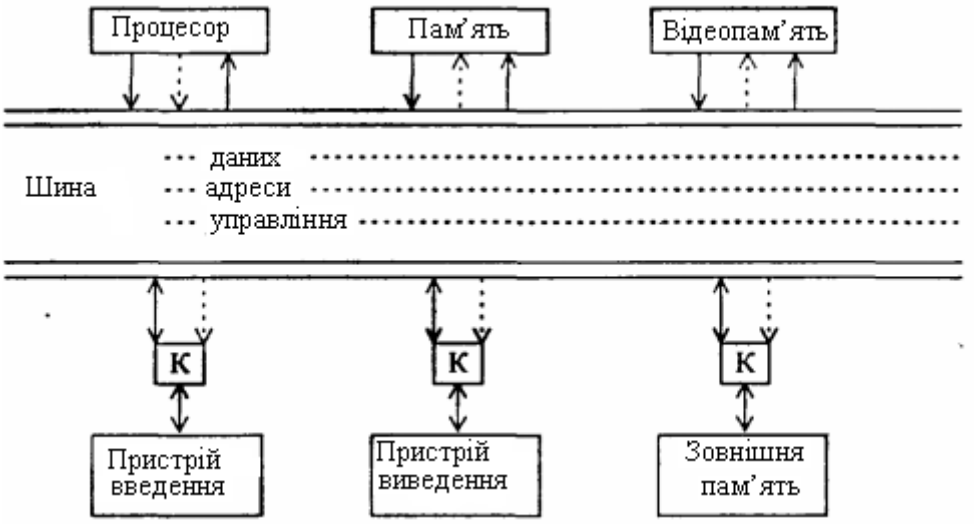

Рис. 2.4 – Шинна (магістральна) архітектура ЕОМ

У наведеній на рис. 2.4 схемі обробку інформації здійснює центральний процесор, синхронізований тактовими імпульсами пристрою синхронізації. Обмін інформацією між блоками здійснюється за трьома магістралями (шинами): адресною, даних і управляючою. Магістраль адреси служить для передачі коду адреси, за якою проводиться звернення до пристроїв введення-виведення та іншими зовнішніми пристроями. Оброблювана інформація та результати обчислень передаються магістраллю даних. Магістраль управління передає управляючі сигнали на всі блоки, налаштовуючи пристрої, що беруть участь у виконанні команди, на потрібний режим роботи.

Кількість ліній, які входять у склад шини, прийнято називати *розрядністю* (*шириною*) цієї шини. Ширина адресної шини, наприклад, визначає максимальний розмір оперативної пам’яті, яка може встановлюватися в обчислювальній системі. Ширина шини даних визначає максимальний об’єм інформації, яка за один такт може одержуватися або передаватися цією шиною.

Використання трьох магістралей забезпечує високу швидкодію і спрощує процес обчислення. Можлива побудова системи з однією або двома магістралями, якими послідовно передаються код адреси і оброблювана інформація, але при цьому значно зростає час виконання команди і ускладнюється організація обміну інформацією між вузлами.

Описану схему легко поповнювати новими пристроями - ця властивість називається ***відкритістю архітектури***. Для користувача відкрита архітектура означає можливість вільно вибирати склад зовнішніх пристроїв для свого комп'ютера, тобто конфігурувати його в залежності від кола розв'язуваних задач.

Теми доповідей

1. Історія розвитку та покоління ЕОМ.
2. Поняття про апаратне та програмне забезпечення та їх взаємодію.
3. Структура та основні функціональні пристрої персональних комп’ютерів. Системний модуль

(блок), монітор, клавіатура, друкуючий пристрій ПК.

Література

1. Зацеркляний М.М. Компютерні основи систем кібербезпеки: навч. посібник/ Зацеркляний М.М., Струков В.М.-Харків: Тов. «В деле», 2017.- 292 с.
2. Інформатика: Комп'ютерна техніка. Комп'ютерні технології: Підручник для студентів вищих навчальних закладів / За ред. Пушкаря О.І. – К.: Каравела, 2004.
4. Основи інформаційних технологій і систем- Підручник / В. А. Павлиш, Л. К. Гліненко, Н. Б. Шаховська. Львів : Видавництво Львівської політехніки, 2018. 620 с

Тема 2 Архітектура обчислювальних систем

Семінарське заняття: Архітектура обчислювальних систем

Мета заняття: Ознайомитись з загальною архітектурою обчислювальних систем

Тривалість: 2 год

Навчальні питання

1. Поняття архітектури.
2. Класифікація архітектур обчислювальних систем.
3. Архітектура фон Неймана.

Поняття архітектури

В обчислювальній науці використовуються три терміни, пов'язані з побудовою електронно-обчислювальної машини. ***Архітектура комп'ютера*** - це опис компонент комп'ютера та їх взаємодії. ***Організація комп'ютера*** - це опис конкретної реалізації архітектури, її втілення "в залізі". Третій термін - це ***схема комп'ютера***, детальний опис його електронних компонент, їх з'єднань, пристроїв живлення, охолодження та інших. Програмісту досить часто потрібне знання архітектури комп'ютера, рідше його організації і ніколи - схеми комп'ютера.

Обчислювальна машина (комп'ютер) - це комплекс технічних і програмних засобів, призначений для автоматизації підготовки і розв'язування задач користувача.

Обчислювальна система - це сукупність взаємопов'язаних процесорів або обчислювальних машин, периферійного обладнання та програмного забезпечення для розв'язування задач користувача.

Основною відмінною рисою обчислювальних систем є наявність в них засобів, що реалізують паралельну обробку за рахунок побудови паралельних гілок обчислення, що як правило, не передбачається в обчислювальних машинах.

Очевидно, що відмінності між обчислювальними машинами та обчислювальними системами не можуть бути точно визначені (обчислювальні машини навіть із одним процесором мають різні засоби розпаралелювання, а обчислювальні системи можуть складатися з традиційних обчислювальних машин або процесорів).

Архітектура обчислювальної системи – це сукупність характеристик і параметрів, що визначають функціонально-логічну і структурно-організовану систему і зачіпають в основному рівень паралельно працюючих обчислювачів. Поняття архітектури охоплює загальні принципи побудови і функціонування, найбільш істотні для користувача.

Революцію в комп'ютерних архітектурах здійснив принцип паралельної обробки даних, що втілює ідею одночасного (паралельного) виконання кількох дій. Обробка даних має два різновиди: конвеєрна та власне паралельна.

Класифікація архітектур обчислювальних систем.

Паралельна обробка. Якщо якийсь пристрій виконує одну операцію за одиницю часу, то тисячу операцій він виконає за тисячу одиниць. Якщо вважати, що є п'ять таких незалежних пристроїв, здатних працювати одночасно, то ту ж тисячу операцій система з п'яти пристроїв може виконати вже не за тисячу, а за двісті одиниць часу. Аналогічно система з N пристроїв ту ж роботу виконає за $1000/N$ одиниць часу. Однак це ідеальне прискорення вдається одержати лише в досить спеціальних ситуаціях, коли підзадачі повністю незалежні (наприклад, задача перебору, злом паролів). Аби стало зрозуміліше, розглянемо приклад з життя. Якщо одна людина скопує один квадратний метр землі за дві хвилини, то це не означає, що дванадцять чоловік скопають його за десять секунд: вони будуть заважати один одному і робота тільки сповільниться. Що ж до обчислень, то є алгоритми, які при розпаралелюванні не тільки не дають прискорення, а навіть сповільнюються.

Конвеєрна обробка. Що необхідно для додавання двох дійсних чисел, поданих у формі з плаваючою комою? Множина дрібних операцій таких, як порівняння порядків, вирівнювання порядків, додавання мантис, нормалізація і т.п. Процесори перших комп'ютерів виконували всі ці "мікрооперації" для кожної пари аргументів послідовно одна за одною до тих пір, поки не доходили до остаточного результату і лише після цього переходили до обробки наступної пари доданків.

Ідея конвеєрної обробки полягає у виділенні окремих етапів виконання загальної операції, причому кожний етап, виконавши свою роботу, передавав би результат наступному, одночасно приймаючи нову порцію вхідних даних. Одержуємо очевидний виграш у швидкості обробки за рахунок суміщення попередньо рознесених у часі операцій.

Мабуть, найбільш ранньою і найбільш відомою є класифікація архітектур обчислювальних систем, запропонована в 1966 році М. Флінном. Класифікація базується на понятті потоку, під яким розуміється послідовність елементів, команд або даних, що обробляється процесором. На основі числа потоків команд і потоків даних Флінн виділяє **чотири класи архітектур: SISD, MISD, SIMD, MIMD.**

SISD (single instruction stream/single data stream) - одиночний потік команд і одиночний потік даних. До цього класу належать, насамперед, класичні послідовні машини, або інакше, машини фоннеймановського типу.

В таких машинах є тільки один потік команд, всі команди обробляються послідовно одна за одною і кожна команда ініціює одну операцію з одним потоком даних.

SIMD (single instruction stream/multiple data stream) - одиночний потік команд і множинний потік даних. В архітектурах подібного роду зберігається один потік команд, що включає, на відміну від попереднього класу, векторні команди. Це дозволяє виконувати одну арифметичну операцію відразу над багатьма даними - елементами вектора. Спосіб виконання векторних операцій не обговорюється, тому обробка елементів вектора може проводитися або процесорною матрицею, як в ILLIAC IV, або за допомогою конвеєра, як, наприклад, в машині CRAY-1.

MISD (multiple instruction stream/single data stream) - множинний потік команд і одиночний потік даних. Передбачається наявність в архітектурі багатьох процесорів, які обробляють один і той же потік даних. Проте ні Флінн, ні інші фахівці в галузі архітектури комп'ютерів до сьогодні не змогли подати переконливий приклад реально існуючої обчислювальної системи, побудованої за даним принципом. Ряд дослідників відносять конвеєрні машини до даного класу, проте це не знайшло остаточного визнання в науковому співтоваристві. Будемо вважати, що поки даний клас порожній.

MIMD (multiple instruction stream/multiple data stream) - множинний потік команд і множинний потік даних. Цей клас допускає, що в обчислювальній системі є кілька пристроїв обробки команд, об'єднаних в єдиний комплекс і працюючих кожний зі своїм потоком команд і даних.

Запропонована схема класифікації аж до сьогодні є застосовною при початковій характеристиці того чи іншого комп'ютера. Якщо говориться, що комп'ютер належить до класу SIMD або MIMD, то відразу стає зрозумілим базовий принцип його роботи, і в деяких випадках цього буває достатньо. Проте видно і явні недоліки. Зокрема, деякі заслуговуючі уваги архітектури, наприклад, dataflow і векторно-конвеєрні машини, чітко не вписуються в дану класифікацію. Інший недолік - це надмірна заповненість класу MIMD. Необхідний засіб, який більш вибірково систематизує архітектури, що за Флінном потрапляють в один клас, але зовсім різниза кількістю процесорів, природою і топологією зв'язку між ними, за способом організації пам'яті і, звичайно ж, за технологією програмування.

На сьогодні існують різні методи класифікації архітектури паралельних обчислювальних систем. Одна із можливих класифікацій полягає в розділенні паралельних обчислювальних систем за типом пам'яті.

Векторно-конвеєрні комп'ютери. Конвеєрні функціональні пристрої і набір векторних команд - це дві особливості таких машин. На відміну від традиційного підходу, векторні команди оперують масивами незалежних даних, що дозволяє ефективно завантажувати доступні конвеєри, тобто команда виду A=B+C може означати додавання двох масивів, а не двох чисел.

Масивно-паралельні комп'ютери (MPP-системи). Масивно-паралельні комп'ютери з розподіленою пам'яттю. Ідея побудови комп'ютерів цього класу тривіальна: візьмемо серійні мікропроцесори, постачимо кожний своєю локальною пам'яттю, з'єднаємо за допомогою деякого комунікаційного середовища - от і все (рис. 2.1).

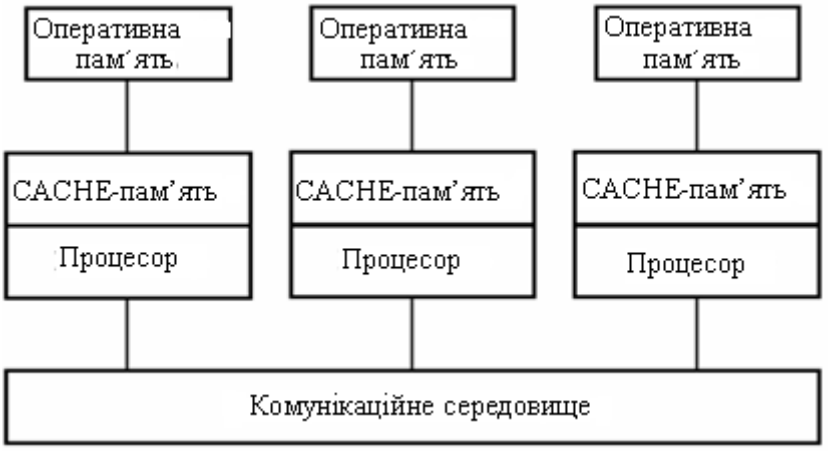

Рис. 2.1. Схема архітектури MPP-системи

Переваг у такої архітектури маса: якщо потрібна висока продуктивність, то можна додати ще процесори; якщо обмежені фінанси або заздалегідь відома необхідна обчислювальна потужність, то легко підібрати оптимальну конфігурацію тощо. Проте є вирішальний "мінус", який зведе багато "плюсів" нанівець. Справа в тому, що міжпроцесорна взаємодія в комп'ютерах цього класу йде набагато повільніше, ніж відбувається локальна обробка даних самими процесорами. Саме тому написати ефективну програму для таких комп'ютерів досить складно, а для деяких алгоритмів - іноді просто неможливо.

До цього ж класу можна віднести і комп'ютерні мережі, які все частіше розглядаються як дешева альтернатива вкрай дорогим суперкомп'ютерам.

Комп'ютери із загальною пам'яттю (SMP-системи). Вся оперативна пам'ять таких комп'ютерів ділиться декількома однаковими процесорами (рис. 2.2). Це знімає проблеми попереднього класу, але додає нові - число процесорів, що мають доступ до загальної пам'яті, з чисто технічних причин не можна зробити великим.

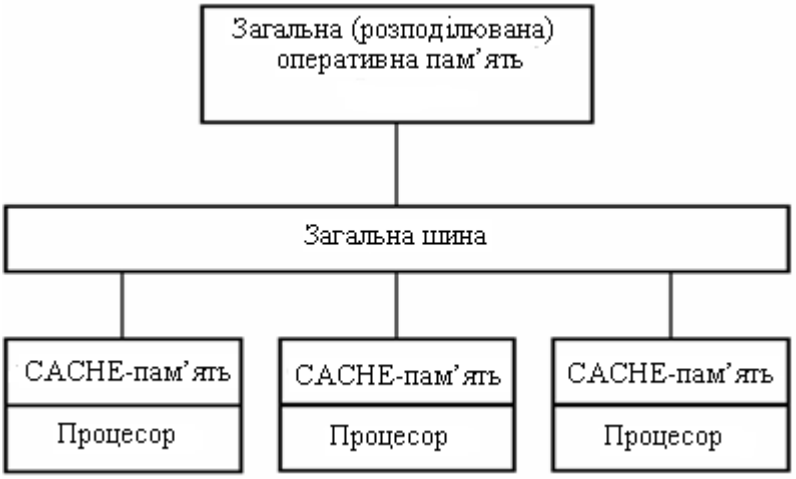

Рис. 2.2. Схема архітектури SMP-системи

Наявність загальної пам'яті значно спрощує взаємодію процесорів між собою, однак за цією уявною простотою ховаються великі проблеми, притаманні системам цього типу. Крім добре відомої проблеми конфліктів при зверненні до загальної шини пам'яті виникла і нова проблема, пов'язана з ієрархічною структурою організації пам'яті сучасних комп'ютерів. Справа в тому, що найвужчим місцем у сучасних комп'ютерах є оперативна пам'ять, швидкість роботи якої значно відстає від швидкості роботи процесора. На сьогодні ***ця швидкість приблизно в 20 разів нижча за необхідну для 100% узгодженості зі швидкістю роботи процесора,*** і розрив весь час збільшується. Аби згладити розрив у швидкості роботи процесора і основної пам'яті, кожний процесор забезпечується ***швидкісною буферною пам'яттю (кеш-пам'яттю),*** що працює зі швидкістю процесора. У зв'язку з цим у багатопроцесорних системах, побудованих на базі таких мікропроцесорів, порушується принцип рівноправного доступу до будь-якої точки пам'яті. Для його збереження доводиться організовувати апаратну підтримку когерентності кеш-пам'яті, що призводить до великих накладних витрат і сильно обмежує можливості з нарощування продуктивності таких систем шляхом простого збільшення числа процесорів.

У чистому вигляді SMP-системи складаються, як правило, не більше ніж із 32 процесорів, а для подальшого нарощування ***використовується NUMA-технологія***, яка на сьогодні дозволяє створювати системи, що включають до декількох сотень процесорів із загальною продуктивністю близько 150 млрд. операцій за секунду. Системи цього типу виробляються багатьма комп'ютерними фірмами як багатопроцесорні сервери з числом процесорів від 2 до 64 і міцно утримують лідерство в класі малих суперкомп'ютерів із продуктивністю до 100 млрд. операцій за секунду.

Системи з неоднорідним доступом до пам'яті (NUMA). NUMA-архітектури є чимось середнім між SMP і MPP. В таких системах пам'ять фізично розподілена, але логічно загальнодоступна. Система складається з однорідних базових модулів (плат), що складаються з невеликого числа процесорів і блоку пам'яті. Модулі об'єднані за допомогою високошвидкісного комутатора. ***Підтримується єдиний адресний простір***, апаратно підтримується доступ до віддаленої пам'яті, тобто до пам'яті інших модулів. При цьому доступ до локальної пам'яті в кілька разів швидший за віддалений.

Комп'ютерні кластери. Кластерні технології стали логічним продовженням розвитку ідей, закладених в архітектурі MPP-систем. Якщо процесорний модуль у MPP-системі є закінченою обчислювальною системою,то наступний крок напрошувався сам собою: чому б за такі обчислювальні вузли не використовувати звичайні комп'ютери, які серійно випускаються. Розвиток комунікаційних технологій, а саме поява високошвидкісного мережевого обладнання та спеціального програмного забезпечення такого, як MPI, що реалізує механізм

передачі повідомлень над стандартними мережевими протоколами, зробило кластерні технології загальнодоступними. Сьогодні не становить великих труднощів створити невелику кластерну систему, об'єднавши обчислювальні потужності комп'ютерів окремої лабораторії або навчального класу.

Привабливою рисою кластерних технологій є те, що вони *дозволяють для досягнення необхідної продуктивності об'єднувати в єдині обчислювальні системи комп'ютери різного типу, починаючи від персональних комп'ютерів і закінчуючи потужними суперкомп'ютерами.* Ці технології використовуються як дешева альтернатива суперкомп'ютерам.

Кластер – це пов'язаний набір повноцінних комп'ютерів, використовуваний як єдиний ресурс. Існує два підходи до створення кластерних систем:

• об'єднання в єдину систему повнофункціональних комп'ютерів, які можуть працювати, в тому числі, і як самостійні одиниці, наприклад, комп'ютери навчального класу або робочі станції лабораторії;

• цілеспрямоване створення потужного обчислювального ресурсу, в якому роль обчислювальних вузлів відіграють комп'ютери, які промислово випускаються, і тоді немає необхідності постачати такі комп'ютери графічними картами, моніторами, дисковими накопичувачами та іншим периферійним обладнанням, що значно здешевлює вартість системи.

В останньому випадку системні блоки комп'ютерів, як правило, компактно розміщуються в спеціальних стійках, а для управління системою і для завантаження задач виділяється один або декілька повнофункціональних комп'ютерів, які називають **хост-комп'ютерами.** Переваги кластерної системи перед набором незалежних комп'ютерів очевидні. По-перше, система пакетної обробки завдань дозволяє послати завдання на обробку кластеру в цілому, а не якомусь окремому комп'ютеру, що дозволяє забезпечити більш рівномірне завантаження комп'ютерів. По-друге, з'являється можливість спільного використання обчислювальних ресурсів декількох комп'ютерів для розв'язування одного завдання.

Для створення кластера використовуються комп'ютери, які можуть бути як простими однопроцесорними системами, так і мати складну архітектуру SMP і навіть NUMA.

Архітектура фон Неймана.

Класична узагальнена структурна схема комп'ютера подається на рис. 2.3. При цьому використовуються такі позначення:

ПУ (пристрій управління) організовує автоматичне виконання програм і функціонування обчислювальної машини. Основне завдання ПУ - вироблення управляючих сигналів і розподіл їх ланцюгом управління.

АЛП (арифметико-логічний пристрій) призначений для виконання арифметичних і логічних операцій над даними, які надходять.

ОЗП (оперативний запам'ятовуючий пристрій) є масивом запам'ятовуючих елементів, організованих у вигляді запам'ятовуючих комірок і здатних зберігати деяку одиницю інформації.

Пристрій введення перетворює вхідну інформацію у послідовність електричних сигналів.

Пристрій виведення перетворює послідовність електричних сигналів у форму, зручну для сприйняття користувачем.

Разом ПУ і АЛП утворюють *ЦП* (центральний процесор).

Концепція фоннеймановської архітектури передбачає один АЛП і одну основну пам'ять. Проте найбільш поширеною на сьогодні є структура обчислювальної системи, що має дві або три (у більшості випадків) загальних магістралі (шини), до яких під впливом пристроїв управління можуть почережно вмикатися вузли, що входять у систему (рис. 2.4).

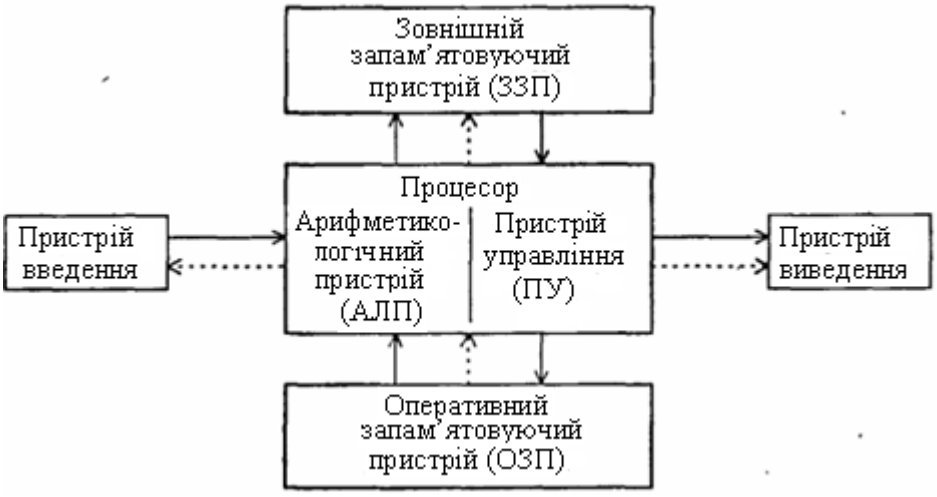

Рис. 2.3 - Архітектура ЕОМ, побудованої на принципах фон Неймана. Суцільні лінії зі стрілками вказують напрям потоків інформації, пунктирні – напрям управляючих сигналів від процесора до решти вузлів ЕОМ.

Теми доповідей

1. Вузли обчислювальних машин
2. Класифікація архітектур обчислювальних систем.
3. Архітектура фон Неймана. .

Література

1. Зацеркляний М.М. Компютерні основи систем кібербезпеки: навч. посібник/ Зацеркляний М.М., Струков В.М.-Харків: Тов. «В деле», 2017.- 292 с.
2. Тарарака В.Д. Архітектура комп'ютерних систем: навчальний посібник. – Житомир : ЖДТУ, 2018. – 383 с.
3. Хіхловська І.В. Обчислювальна техніка та мікропроцесори : підручник / Хіхловська І.В., Антонов О.С. – Одеса: ОНАЗ ім. О.С. Попова, 2011. – 440 с

Тема 2 Архітектура обчислювальних систем

Семінарське заняття: Архітектура процесорів

Мета заняття: Ознайомитись з загальною архітектурою процесорів

Тривалість: 2 год

Навчальні питання

1. Узагальнена схема центрального процесора.
2. Пристрій управління
3. Арифметико-логічний пристрій
4. Процесори з повним і скороченим набором команд.
5. Процес передачі команди від процесора до зовнішнього пристрою
6. Реалізація програми центральним процесором

1. Узагальнена схема центрального процесора

Аби мати уяву про структуру та функції процесора, звернемося до схеми гіпотетичної ЕОМ, модель якої традиційно називається фон-нейманівською (рис. 2.3).

Пристрій управління (ПУ) організує автоматичне виконання програм і функціонування ЕОМ як єдиної системи. Основне завдання ПУ - вироблення управляючих сигналів (УС) і розподіл їх ланцюгами управління.

Арифметично-логічний пристрій (АЛП) призначений для виконання арифметичних і логічних операцій над даними, які в нього поступають.

Оперативна пам'ять (ОП) є масивом запам'ятовуючих елементів (ЗЕ), організованих у вигляді комірок, здатних зберігати певну одиницю інформації.

Процесором називається функціональна частина ЕОМ, призначена для безпосереднього здійснення процесу перетворення (обробки) інформації та управління цим процесом. Іншими словами - це сукупність арифметико-логічного пристрою і пристрою управління.

На рис. 2.5 приведена функціональна структура гіпотетичного процесора.

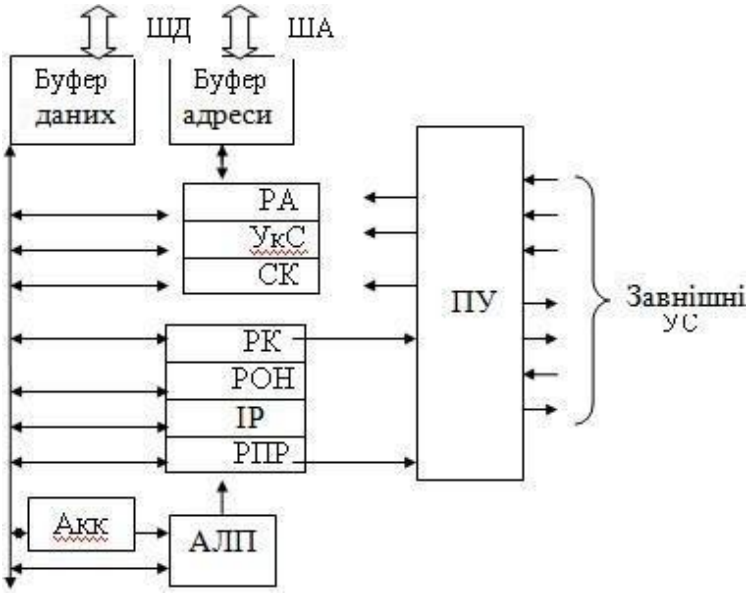

Рис. 2.5. Структура гіпотетичного процесора

Регістр адреси (РА) призначений для зберігання адреси комірки основної пам'яті аж до завершення операції (зчитування або запис) з цією комірки.

Покажчик стека (УкС) - це регістр, де зберігається адреса вершини стека. У реальних обчислювальних машинах стек реалізується у вигляді ділянки основної пам'яті, як правило, розташованої в області найбільших адрес. Заповнення стека відбувається в бік зменшення адрес, при цьому вершина стека - це комірка, куди проведений останній за часом запис. Для зберігання адреси такої комірки призначений УкС. При виконанні операції занесення в стек вміст УкС спочатку зменшується на одиницю, після чого використовується як адреса, за якою проводиться запис. Відповідна комірка стає новою вершиною стека. Зчитування зі стека відбувається з комірки, на яку вказує поточна адреса в УкС, після чого вміст покажчика стека збільшується на одиницю. Таким чином, вершина стека опускається, а прочитане слово вважається видаленим зі стека. Хоча фізично прочитане слово і залишилося в комірці пам'яті, при наступному записі в стек воно буде замінено новою інформацією.

Лічильник команд (СК) - невід'ємний елемент процесора будь-якої ЕОМ, побудованої відповідно до фон-нейманівського принципу програмного управління. Згідно з цим принципом сусідні команди програми розташовуються в комірках пам'яті з наступними за порядком адресами і виконуються переважно в тій же черговості, у якій вони розміщені в пам'яті ЕОМ. Таким чином, адреса чергової команди може одержуватися шляхом збільшення адреси комірки, з якої зчитана поточна команда, на довжину виконуваної команди, подану числом займаних нею комірок. Реалізацію такого режиму і забезпечує лічильник команд - двійковий лічильник, в якому зберігається і модифікується адреса чергової команди програми. Перед початком обчислень в СК заноситься адреса комірки основної пам'яті, де зберігається команда, яка повинна виконуватися першою. В процесі виконання кожної команди шляхом збільшення вмісту СК на довжину виконуваної команди в лічильнику формується адреса наступної команди, яка підлягає виконанню. В даному випадку будь-яка команда займає одну комірку, тому вміст СК збільшується на одиницю. По завершенні поточної команди адреса наступної команди програми завжди береться з лічильника команд. Для зміни природного порядку обчислень (переходу в іншу точку програми) достатньо занести в СК адресу точку переходу.

Лічильник команд визначає лише місце розташування команди в пам'яті, але не містить інформації про те, що це за команда. Аби приступити до виконання команди, її необхідно дістати з пам'яті і розмістити в регістрі команди (РК). Цей етап має назву *вибірки команди*. Тільки з моменту завантаження команди в РК вона стає «видимою» для процесора. У РК команда зберігається протягом усього часу її виконання.

Регістри загального призначення (РОН) служать для тимчасового зберігання операндів і результатів обчислень. Це найшвидший і мінімальний за ємністю тип пам'яті, який іноді об'єднують поняттям надоперативний запам'ятовуючий пристрій - СОЗУ. Як правило, кількість регістрів невелика, хоча в архітектурах зі скороченим набором команд їх число може доходити до декількох десятків.

Індексні регістри (ІР) служать для формування адрес операндів при реалізації циклічних ділянок програм.

Регістр ознаки результату (РПР) призначений для фіксації та зберігання ознаки, що характеризує результат останньої виконаної арифметичної або логічної операції. Такі ознаки можуть інформувати про рівність результату нулю, про знак результату, про виникнення переносу зі старшого розряду, переповнення розрядної сітки тощо. Вміст РПР, як правило, використовується пристроєм управління для реалізації умовних переходів за результатами операцій АЛУ. Під кожну із можливих ознак відводиться один розряд РПР.

Акумулятор (Акк) - це регістр, на який покладаються найрізноманітніші функції. Так, в нього попередньо завантажується один із операндів, що беруть участь в арифметичній або логічної операції. В Акк може зберігатися результат попередньої команди і в нього ж заноситься результат чергової операції. Через Акк часто проводяться операції введення та виведення. Акумулятор можна віднести як до АЛУ, так і до ПУ, а в ЕОМ з реєстровою архітектурою його можна розглядати як один із регістрів загального призначення.

Буфер даних покликаний компенсувати різницю у швидкодії запам'ятовуючих пристроїв і пристроїв, що виступають у ролі джерел і споживачів інформації. У буфер даних при читанні заноситься вміст комірки ОП, а при запису - розміщується інформація, що підлягає збереженню в комірці ОП.

Наявність буфера адреси також дозволяє компенсувати відмінності у швидкодії оперативної пам'яті та інших пристроїв ЕОМ.

2. Пристрій управління

Процес функціонування ЕОМ складається з послідовності елементарних дій у її вузлах. Елементарні дії над інформацією, виконувані протягом одного такту сигналів синхронізації, називаються ***мікрооперaціями*** (МО). Сукупність управляючих сигналів, що викликають одночасно виконувані мікрооперації, утворює ***мікрокоманду*** (МК). У свою чергу, послідовність мікрокоманд, визначену змістом і порядком реалізації машинного циклу, прийнято називати мікропрограмою. Управляючі сигнали виробляються пристроєм управління, а точніше одним із його вузлів - мікропрограмним автоматом (МПА). Назва відображає те, що МПА визначає мікропрограму як послідовність виконання мікрокоманд.

Мікропрограми ініціюються обладнанням, яке виробляє необхідну послідовність управляючих сигналів і входить до складу управляючої частини пристрою управління. Виконуються мікропрограми виконавчим

обладнанням, що входить до складу основної пам'яті, арифметико-логічного пристрою та інших пристроїв ЕОМ.

В узагальненій структурі пристрою управління (рис. 2.6) можна виділити дві частини: управляючу та адресну.

Управляюча частина пристрою управління призначена для координування роботи операційного блоку АЛП ЕОМ, адресної частини пристрою управління, основної пам'яті та інших вузлів ЕОМ. ***Адресна частина*** пристрою управління забезпечує формування адреси команд і виконавчих адрес операндів в основній пам'яті.

Регістр команди РК призначений для приймання чергової команди із запам'ятовуючого пристрою.

Мікропрограмний автомат МПА на підставі результатів розшифрування операційної частини команди (коду операції КОП і лічильника адрес ЛА) виробляє певну послідовність мікрокоманд, що викликають виконання усіх цільових функцій пристрою управління.

Вузол переривань і пріоритетів ВПП дозволяє реагувати на різні ситуації, пов'язані як із виконанням робочих програм, так і зі станом ЕОМ в цілому.

Операційний вузол пристрою управління ***ОВПУ***, який називається інакше вузлом індексної арифметики або вузлом адресної арифметики, обробляє адресні частини команд, формуючи виконавчі адреси операндів, а також готує адресу наступної команди при виконанні команд переходу. Склад ОВПУ може бути аналогічний складу основного операційного пристрою ЕОМ (іноді в найпростіших обчислювальних машинах із метою економії витрат на обладнання ОВПУ поєднується з основним операційним пристроєм).

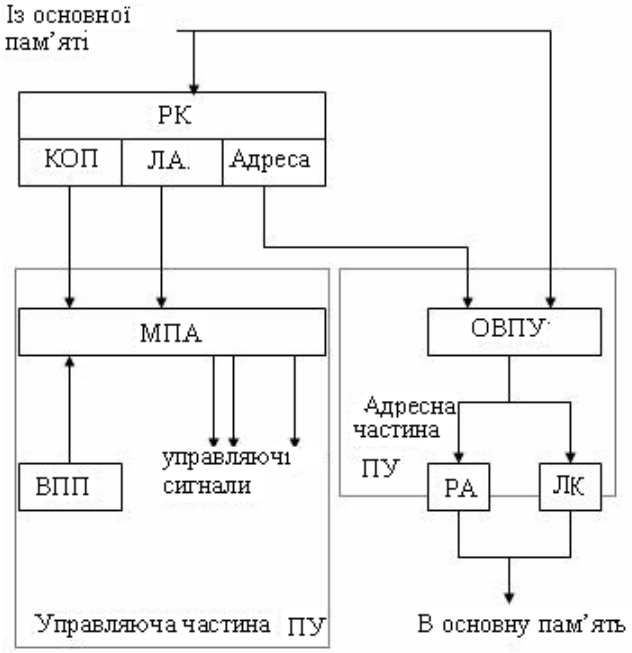

Рис. 2.6. Узагальнена структура пристрою управління

Регістр адреси РА використовується для зберігання виконавчих адрес операндів. Лічильник команд СК необхідний для вироблення і зберігання адрес команд.

Вміст РА і СК посилається в регістр адреси основної пам'яті (ОП) для вибірки операндів і команд відповідно. До складу ПУ можуть також входити додаткові вузли, зокрема ***вузол організації прямого доступу до пам'яті***. Цей вузол, як правило, реалізується у вигляді самостійного пристрою - контролера прямого доступу до пам'яті (КПДП). КПДП забезпечує суміщення в часі роботи центрального процесора з процесом обміну інформацією між ОП та іншими периферійними пристроями (ПФП) ЕОМ, тим самим підвищуючи загальну продуктивність машини.

Основним вузлом будь-якого пристрою управління є МПА. Всі множини технологій, що використовуються при реалізації мікропрограмних автоматів пристроїв управління, можна звести до двох категорій:

- МПА з «жорсткою» логікою чи апаратною реалізацією;
- МПА з програмованою логікою.

Пристрій управління з жорсткою логікою має у своєму складі такий мікропрограмний автомат, вихідні сигнали якого виробляються за рахунок з'єднання між собою логічних схем. Вихідною інформацією є прапорці, що містять команди, тактові імпульси і сигнали, які надходять із шини управління. Код операції, що зберігається в регістрі команд, використовується для визначення того, які керуючі сигнали і в якій послідовності повинні формуватися.

Дешифратор коду операції перетворює код *j*-ої операції в одиничний сигнал на *j*-му виході

дешифратора.

Машинний цикл виконання кожної команди складається з декількох тактів. Сигнали управління виробляються в строго певні моменти часу, тобто вони прив'язані до імпульсів синхронізації. Процес синтезу мікропрограмного автомата з жорсткою логікою називається структурним синтезом, який має такі етапи:

- вибір типу логіки і запам'ятовуючих елементів;
- кодування станів автомата;
- синтез комбінаційної схеми.

Кожній мікропрограмі відповідає свій набір логічних схем із фіксованими зв'язками між ними. Такі автомати економічні, мають найбільшу швидкодію, але зі зростанням складності ускладнюються і автомат важко реалізувавти. До того ж автомат має нульову регулярність мікропрограмної схеми і незмінність.

Для ініціювання мікрооперації достатньо сформувати відповідний керуючий сигнал на відповідній шині управління. Це переводить лінію в активний стан. Для вказівки мікрооперації, що виконується на даному такті, можна сформувати управляюче слово, в якому кожний біт відповідає одній лінії. Таке управляюче слово називається мікрокомандою. Таким чином мікрокоманда - це послідовність нулів і одиниць. Послідовність мікрокоманд, що реалізують певний етап машинного циклу називається мікропрограмою. Мікропрограма розміщується в спеціальному пристрої пам'яті - пам'яті мікрокоманд.

Причина популярності автоматів із програмованою логікою в тому, що вони допускають розробку досить складних і взаємозалежних мікропрограм і допускають внесення змін. Кожній команді обчислювальної машини в пам'яті мікрокоманд відповідає мікропрограма.

Завантаження мікропрограм, що виконують операції, здійснюється шляхом передачі коду оперативної пам'яті з регістра команди на вхід перетворювача коду операції. У перетворювачі код операції трансформується на адресу першої мікрокоманди. Ця адреса надходить у формувач адреси мікрокоманди і далі в регістр адреси мікрокоманд.

Мікрооперaційна частина команди (МО) надходить ***на дешифратор мікрооперації*** (ДШМО). На його виході утворюються управляючі сигнали. Саме ці сигнали ініціюють виконання мікрооперацій у виконуючих пристроях і вузлах обчислювальних машин.

Адресна частина мікрокоманди подається у формувач адреси мікрокоманди. Адреса наступної мікрокоманди формується на основі 3-х частин:

- перетворювача коду операції;
- значень інформуючих сигналів прапорців;
- адресної частини.

3. Арифметико-логічний пристрій

Арифметико-логічний пристрій (АЛП) - це пристрій, призначений для арифметичної і логічної обробки даних. Він у загальному випадку виглядає так, як показано на рис. 2.7.

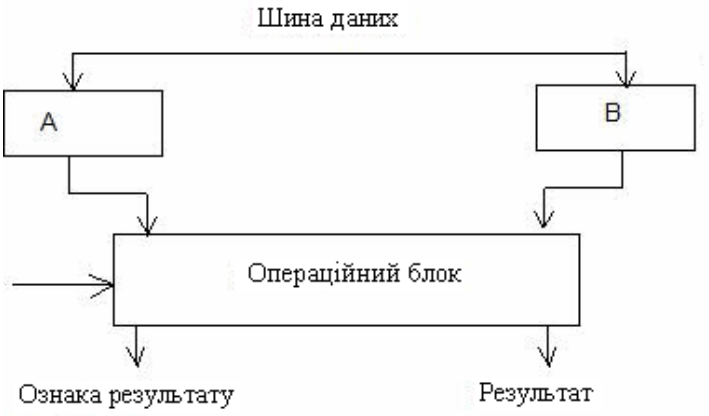

Рис. 2.7

Операційний блок є тією частиною АЛП, яка виконує арифметичні і логічні операції. Вибір конкретної операції з можливого списку операцій виконується кодом операції команди, що зберігається в даний момент у регістрі команд. Після перетворення коду операції в мікропрограмному автоматі останній управляючий сигнал надходить в операційний блок АЛП. Операційний блок будується як сукупність комбінаційних схем, тобто операційні блоки не мають внутрішньої пам'яті. А це означає, що до моменту збереження результату операнди (вхідні дані) повинні бути присутніми на вході блоку. На рис. 2.8 подається схема операційного блоку.

На рисунку 2.7 регістр *A* зберігає перший операнд, регістр *B* - другий операнд до запису їх в оперативну пам'ять.

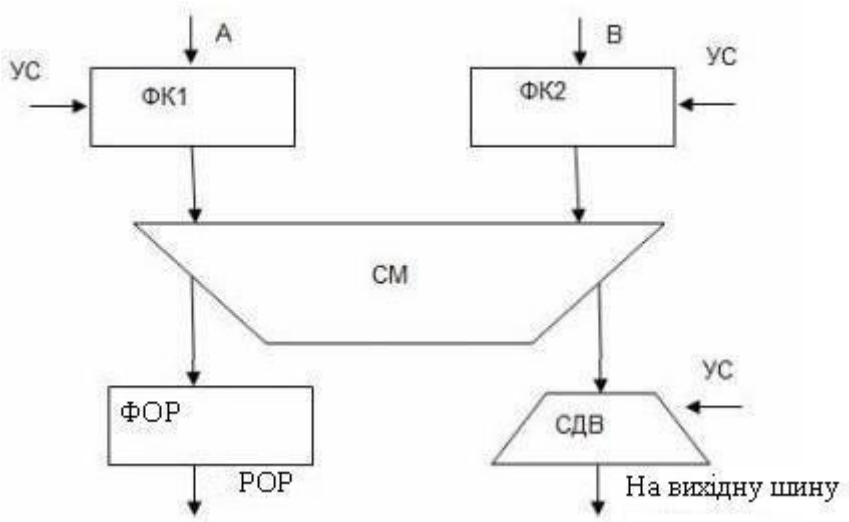

Рис. 2.8. Узагальнена схема операційного блоку

На рис. 2.8 введені такі позначення:
ФК - формувач коду;
СМ – суматор;
ФОР - формувач ознаки результату;
СДВ – зсувач;
УС - управляючі сигнали;
РОР – регістр ознаки результату.

Суматор СМ виконує операції арифметичного додавання, додавання за модулем, логічного додавання, логічного множення.

Формувач ознаки результату ФОР виробляє інформаційні сигнали, передані в пристрій управління: ознака знаку, ознака переповнення, ознака нульового значення.

Зсувач СДВ служить для виконання мікрооперацій зсуву на виході суматора, якщо це потрібно.

Розрізняють 2 типу операційних блоків:

1) паралельний (всі розряди обробляються одночасно і внутрішні переноси реалізуються внутрішніми схемами операційного блоку);

2) послідовний (існують однорозрядні процеси).

Процесори з повним і скороченим набором команд

Невідповідність стандартів підходів, принципів мови високого рівня і обчислювальної машини Фон-Неймана та мови низького рівня називається семантичним розривом. Подолати такий розрив намагаються шляхом розширення системи команд, доповнюючи цю систему командами, які реалізують складні оператори мови високого рівня, але на апаратному рівні. Обчислювальні машини, де реалізовані такі засоби, тобто ***машини з повним набором команд*** називаються - ***CISC*** (Complex Instruction Set Computer).

Спроби за допомогою CISC подолати семантичний розрив призводять до ускладнення архітектури обчислювальної машини, особливо пристрою управління, що позначається на продуктивності машини в цілому. У CISC складно організувати ефективний конвеєр команд, який є одним із найбільш перспективних шляхів збільшення продуктивності обчислювальної машини. Аналіз програм, одержаних після компіляції з мови високого рівня, визначив такі закономірності: реалізація складних команд, еквівалентних операторам мови високого рівня, вимагає збільшення ємності управління пам'яттю в мікропрограмному пристрої управління. Мікропрограми складних команд можуть займати до 60% обсягу ПЗП.

У скомпільованій програмі оператори мови високого рівня організовані у вигляді процедур, тому на виконання виклику процедури і повернення з неї припадає від 15% до 45% обчислювального навантаження. При виклику процедури викликаюча програма передає їй деяку кількість аргументів. У 98% випадків їх число не перевищує 6-ти. Приблизно таке ж становище складається і з параметрами, які процедура повертає. Більше 80% змінних, використовуваних програмою, локальні.

Майже половину операцій у ході обчислень становить операція присвоювання. Аналіз наведених результатів вимагає перегляду традиційних архітектурних рішень, наслідком чого є поява RISC-архітектури.

RISC (Reduced Instruction Set Computer) характерна тим, що регістрів стає багато, команди мають однакову довжину (для конвеєра), уніфіковані канали передачі.

Головні зусилля в RISC спрямовані на побудову ефективного конвеєра команд. Всі команди дістаються з оперативної пам’яті, при цьому жодна команда не повинна перебувати в стані очікування. Ідеальним вважається варіант, коли будь-який етап циклу команди виконується протягом одного такту. Цю умову можна реалізувати для етапу вибірки. Для цього необхідно, аби всі команди мали стандартну довжину.

Уніфікація часу виконання команд більш складне завдання, тому разом зі зверненнями до регістрів існує звернення до ОП. Однакова довжина команди - це не все. Скорочення самого набору команд. У в 80%-90% випадків використовується 10%-20% команд від загального списку. До найбільш часто використовуваних дій відноситься пересилання даних, а також алгебраїчні і логічні операції. Скоротимо число команд, що мають доступ до ОП. Для цього збільшимо кількість регістрів загального призначення.

Концепція RISC архітектури зводиться до таких положень:

• виконання всіх або принаймні 75% команд за 1 цикл;

• стандартна однослівна довжина всіх команд, що дорівнює природній довжині слова даних (операндів) і ширині ШД (це допускає уніфіковану потокову обробку всіх команд);

• кількість команд не більше 128-ми (реально 70);

• мала кількість форматів команд (не більше 4-ох).

Команди використовують внутрішньопроцесорні міжрегістрові пересилання. Регістрів загального призначення мінімум 32, максимум більше 500.

До *переваг RISC* відносяться:

1. Порівняно проста структура ПУ, яка в результаті займає в RISC-процесорі не більше 10% площі кристала.

2. Висока швидкодія, пов'язана з уніфікацією набору команд і орієнтацією на потокову конвеєрну обробку.

При цьому із-за скорочення числа команд на виконання ряду функцій доведеться витрачати кілька команд. Це подовжує код команди. У середньому довжина однієї і тієї ж команди у RISC-архітектурі більша за довжину аналогічної команди у CISC в середньому на 30%. Велика кількість регістрів загального призначення. Зі збільшенням числа регістрів ускладнюється процедура їх адресації. ПУ - пристрій із жорсткою логікою.

2. Процес передачі команди від процесора до зовнішнього пристрою

Взаємодія центрального процесора із зовнішніми пристроями передбачає виконання логічної послідовності дій, пов'язаних із пошуком пристрою, визначенням його технічного стану, обміном командами та інформацією. Ця логічна послідовність дій разом із пристроями, які її реалізують, одержала назву ***інтерфейс введення-виведення***.

Для різних пристроїв можуть використовуватися різні логічні послідовності дій, тому інтерфейсів введення-виведення може в одній і тій же ЕОМ використовуватися декілька. Якщо їх вдається звести до одного, універсального, то такий інтерфейс називається стандартним. У IBM PC є три стандартних інтерфейсу для зв'язку центрального процесора із зовнішніми пристроями: ***паралельний*** (типу **Centronics**) і два ***послідовних*** (типу **RS-232** і **USB**).

Процес передачі команди до пристрою такий:

• процесор виставляє на шину адреси адресу пристрою, на шину управління - сигнал «пошук пристрою»;

• пристрій відгукується на шину управління сигналом «збіг адреси»;

• процесор спеціальним сигналом запитує байт стану і одержує його;

• процесор розміщує на шину даних команду і сигнал на шину управління «передача команди»;

• процесор очікує від пристрою підтвердження про приймання команди. І після одержання переходить до виконання чергової команди.

Якщо при зверненні центрального процесора до зовнішнього пристрою продовження виконання основної програми центральним процесором можливе тільки після завершення операції введення-виведення, то центральний процесор, завантаживши зовнішній пристрій, переходить у стан очікування і знаходиться в ньому до тих пір, поки зовнішній пристрій не повідомить йому про закінчення обміну даними. Це призводить до простою більшості пристроїв ЕОМ, оскільки в кожний момент часу може працювати тільки один із них. Такий ***режим роботи*** одержав назву ***однопрограмного*** - в кожний момент часу всі пристрої знаходяться в стані очікування і тільки один пристрій виконує основну (і єдину) програму.

Для ліквідації таких простоїв і підвищення ефективності роботи обладнання зовнішні пристрої виконані автономними. Одержавши від центрального процесора необхідну інформацію, вони самостійно організовують свою роботу з обміну даними. Процесор, завантаживши зовнішній пристрій, намагається продовжити виконання програми. В разі необхідності (якщо зустрінуться відповідні команди) він може завантажити в роботу кілька інших пристроїв (оскільки зовнішні пристрої працюють значно повільніше процесора). Якщо ж йому доводиться переходити в режим очікування, то, користуючись тим, що в оперативній пам’яті може одночасно перебуватися не одна, а кілька програм, центральний процесор переходить до виконання чергової програми. При цьому створюється ситуація, коли в один і той же момент часу різні пристрої ЕОМ виконують або різні програми, або різні частини однієї і тієї ж програми. Такий ***режим роботи*** ЕОМ називається ***багатопрограмним***.

3. Реалізація програми центральним процесором

Програма в ЕОМ реалізується центральним процесором за допомогою послідовного виконання команд, що утворюють цю програму. Дії, необхідні для вибірки (діставання з основної пам'яті) і виконання команди, називається циклом команди. У загальному випадку цикл команди включає в себе кілька складових (етапів):

• вибірку команди;

• формування адреси наступної команди;
• декодування команди;
• обчислення адрес операндів;
• вибірку операндів;
• виконання операції;
• формування ознаки результату;
• запис результату.

Перераховані етапи виконання команди називаються ***стандартним циклом команди***. Зазначимо, що не всі етапи присутні при виконанні будь-якої команди (залежить від типу команди), проте етапи вибірки, декодування, формування адреси наступної команди і виконання операції мають місце завжди.

У певних ситуаціях можливі ще два етапи:
• непряма адресація;
• реакція на переривання.

Коротко охарактеризуємо кожний із перерахованих етапів стандартного циклу команди. Тут варто враховувати, що опис, який приводиться, має на меті лише дати уявлення про сутність кожного з етапів. У той же час розподіл функцій за різними етапами циклу команди і послідовність виконання деяких з них у реальних ЕОМ можуть відрізнятися від викладеного.

Цикл будь-якої команди розпочинається з того, що центральний процесор дістає команду з пам'яті, використовуючи адресу, що зберігається в лічильнику команд. Двійковий код команди міститься в регістрі команди і з цього моменту стає «видимим» для процесора. Якщо довжина команди збігається з розрядністю комірки пам'яті, то все зрозуміло. Проте, система команд багатьох ЕОМ передбачає кілька форматів команд, причому в різних форматах команда може займати 1, 2 чи більше комірок, а етап вибірки команди можна вважати завершеним лише після того, як в регістрі команди розміщується повний код команди. Інформація про фактичну довжину команди міститься в полях коду операції та способу адресації. Як правило, ці поля розташовуються у першому слові коду команди і для з'ясування необхідності продовження процесу вибірки необхідне попереднє декодування їх вмісту. Таке декодування може бути виконаним після того, як перше слово коду команди опиниться в регістрі команд. У разі багатослівного формату команди процес вибірки триває аж дозанесення в регістр команд усіх слів команди.

Для більшості ЕОМ характерне розміщення сусідніх команд програми в суміжних комірках пам'яті. Якщо взята команда не порушує природного порядку виконання програми, то для обчислення адреси наступної виконуваної команди достатньо збільшити вміст лічильника команд на довжину поточної команди, подану кількістю зайнятих кодом команди комірок пам'яті. Довжина команди, а також те, чи здатна вона змінити природний порядок виконання команд програми, з'ясовуються в ході попереднього декодування. Якщо взята команда здатна змінити послідовність виконання програми (команда умовного чи безумовного переходу, виклику процедури тощо), процес формування адреси наступної команди переноситься на етап виконання операції. Із-за цього у ряді ЕОМ розглянутий етап циклу команди знаходиться не за вибіркою команди, а в кінці циклу.

Після вибірки команда декодується, для чого центральний процесор розшифровує код команди, що знаходиться в регістрі команд. В результаті декодування з'ясовуються такі питання:
• чи знаходиться в регістрі команд повний код команди чи потрібне дозавантаження інших слів команди;
• які подальші дії потрібні для виконання даної команди;
• якщо команда використовує операнди, то звідки вони повинні бути взяті (номер регістра або адресу комірки основної пам'яті);
• якщо команда формує результат, то куди цей результат повинен бути спрямований.

Відповіді на два перших запитання дає розшифровка коду операції, результатом якої може бути унітарний код, де кожний розряд відповідає одній із команд. На практиці замість унітарного коду можуть зустрітися найрізноманітніші форми подання результатів декодування, наприклад адреса комірки спеціальної управляючої пам'яті, де зберігається перша мікрокоманда мікропрограми для реалізації зазначеної в команді операції.

Повне з'ясування всіх аспектів команди, крім розшифровки коду операції, вимагає також аналізу адресної частини команди, включаючи поле способу адресації.

За результатами декодування проводиться підготовка електронних схем ЕОМ до виконання запропонованих командою дій.

Етап обчислення адрес операндів має місце, якщо в процесі декодування команди з'ясовується, щокоманда використовує операнди. Якщо операнди розміщуються в основній пам'яті, здійснюється обчислення їх виконавчих адрес з урахуванням зазначеного в команді способу адресації. Так, у разі індексної адресації для одержання виконавчої адреси проводиться підсумовування вмісту адресної частини команди і вмісту індексного регістра.

Обчислені виконавчі адреси використовуються для зчитування операндів із пам'яті і занесення їх у певні регістри процесора. Наприклад, у разі арифметичної команди операнд після діставання з пам'яті може бути завантаженим у вхідний регістр арифметико-логічного пристрою. Проте частіше операнди попередньо заносяться в спеціальні допоміжні регістри процесора, а їх пересилання на вхід арифметико-логічного пристрою відбувається на етапі виконання операції.

На етапі виконання операції реалізується зазначена в команді операція. Із-за відмінності сутності кожної команди ЕОМ зміст цього етапу суто індивідуальний.

На етапі формування ознаки результату визначається, яким дістався результат операції. Результат може бути додатним, від'ємним, рівним нулю і т.п. Сформована ознака заноситься в регістр ознаки результату для подальшого використання пристроєм управління.

Етап запису результату присутній у циклі тих команд, які допускають занесення результату в регістр або комірку основної пам'яті. Фактично його можна вважати частиною етапу виконання, особливо для тих команд, які розміщують результат відразу в кілька місць.

Чимало команд допускають читання операндів із пам'яті або запис у пам'ять. У найпростішому випадку в адресному полі таких команд явно вказується виконавча адреса відповідної комірки ОП. Проте часто використовується й інший спосіб вказівки адреси, коли адреса операнда зберігається в якійсь комірці пам'яті, а в команді вказується адреса комірки, що містить адресу операнда. Подібний прийом називається ***непрямою адресацією***. Аби прочитати або записати операнд, спочатку потрібно дістати з пам'яті його адресу і тільки після цього провести потрібну дію (читання або запис операнда), іншими словами, потрібно виконати два звернення до пам'яті. Це, природно, відображається і на циклі команди, в якому з'являється непряма адресація. Етап непрямої адресації можна віднести до етапу обчислення адрес операндів, оскільки його сутність зводиться до визначення виконавчої адреси операнда. Іншими словами, вміст адресного поля команди в регістрі команд використовується для звернення до комірки ОП, в якій зберігається адреса операнда, після чого одержана з пам'яті виконавча адреса операнда розміщується в адресне поле регістра команди на місце непрямої адреси. Подальше виконання команди протікає стандартним чином.

Спосіб адресації – це найважливіша характеристика архітектури обчислювальної машини, яка дозволяє або не дозволяє вирішити такі суперечності: з одного боку (з точки зору скорочення апаратних витрат) прагнення зменшити довжину адресного поля, з іншого боку спосіб задання адреси повинен сприяти максимальному зближенню конструктивних характеристик мов програмування високого рівня і машинних команд. Вирішення цієї суперечності і призвело до використання різних способів адресації.

При ***безпосередній адресації*** в адресному полі міститься сам операнд. Застосовується такий спосіб при виконанні арифметичних операцій, операцій порівняння, завантаження констант у регістри. Перевагою такого способу адресації є швидкість виконання, оскільки немає звернення в оперативну пам'ять. Недоліки такого способу: розмір операнда обмежений довжиною адресного поля команди.

При ***прямій адресації*** адресний код ($A_к$) прямо вказує номер комірки пам'яті, до якої проводиться звернення. Виконавчий код ($A_{вик}$) збігається з адресним кодом. Недоліком такого способу адресації є недостатні можливості при зверненні до адресного простору великого розміру. Адреса вказана в команді не може бути зміненою в процесі обчислень.

Непряма адресація – це адресація адреси. Її переваги: дозволяє подолати труднощі, які є у прямій і безпосередній адресації. Недолік полягає у подвійному зверненні до пам'яті (погіршує часові характеристики, витрачається 2 комірки оперативної пам'яті).

Регістрова адресація нагадує пряму адресацію. Різниця в тому, що адресне поле вказує не на комірку пам'яті, а на регістр процесора. Як правило, розмір адресного поля дорівнює 3 і 4 бітам, що дозволяє адресувати 8 і 16 регістрів загального призначення. Переваги реєстрової адресації такі:

1) коротке адресне полі команди;

2) виключаються звернення до оперативної пам'яті.

Недоліки: Обмежене число регістрів загального призначення не дозволяє широко використовувати цю адресацію.

При ***непрямій регістровій адресації*** виконавча адреса операнда зберігається не в комірці основної пам'яті, а в регістрі процесора. Переваги: такі ж можливості, як і у непрямій адресації, але на одне звернення до пам'яті менше. Недоліки: обмежене число регістрів загального призначення не дозволяє широко використовувати цю адресацію.

При ***відносній адресації*** виконавча адреса визначається сумою адресного коду і деякого числа, яке називається базовою адресою. Для зберігання базових адрес в обчислювальній машині можуть передбачатися спеціальні регістри або виділені комірки пам'яті.

У команді в адресному полі виділяється підполе для зазначення номера базового регістра. У ньому знаходиться адреса першої комірки масиву. Залишена частина адресного поля містить адресний код, який використовується для подання порівняно короткого зміщення відносно початку масиву.

Найчастіше виконавча адреса при базуванні утворюється за допомогою суматора. Відносна адресація використовується для доступу до елементів масиву, знаходження якого в пам'яті в процесі обчислень може змінюватися. Зміщення має меншу довжину, ніж повна адреса, і це дозволяє скоротити довжину адресного поля команди, а отже, і саму команду.

Індексна адресація **.** Характерним для реалізації в машині розв'язування математичних задач є циклічність обчислювальних процесів. Це означає, що одна і та ж команда виконується, але над різними операндами, розташованими впорядковано в пам'яті.

Програмування обчислювальних циклів спрощується, якщо після кожного циклу забезпечується автоматична зміна відповідно команд і їх адресних частин. Причому зміна така, що узгоджується з розташуванням у пам'яті виконавчих операндів. Індексна адресація є зручним механізмом для організації

циклічних обчислень.

Теми доповідей

1. Процесори Intel
2. Процесори AMD
3. RISC-процесори

Література

.

1. Зацеркляний М.М. Компютерні основи систем кібербезпеки: навч. посібник/ Зацеркляний М.М., Струков В.М.-Харків: Тов. «В деле», 2017.- 292 с.
2. Тарарака В.Д. Архітектура комп'ютерних систем: навчальний посібник. – Житомир : ЖДТУ, 2018. – 383 с.
3. Хіхловська І.В. Обчислювальна техніка та мікропроцесори : підручник / Хіхловська І.В., Антонов О.С. – Одеса: ОНАЗ ім. О.С. Попова, 2011. – 440 с

Тема 3 Компютерні мережі

Семінарське заняття: Основи комп'ютерних мереж

Мета заняття: Ознайомитись з основами побудови та функціонування комп'ютерних мереж

Тривалість: 2 год

Навчальні питання

1. Класифікація і склад комп'ютерних мереж.
2. Компоненти комп'ютерних мерех
3. Технічне забезпечення ЛОМ
4. Кабелі мереж
5. Мережеві топології

1. Класифікація і склад комп'ютерних мереж

Комп'ютерна мережа – це сукупність комп'ютерів, об'єднаних каналами передачі даних для обміну апаратними, програмними та інформаційними ресурсами. Комп'ютерні мережі виникли як логічний наслідок гострої необхідності розв'язання наступних проблем:

1) обмежений обсяг ресурсів – оперативної і зовнішньої пам'яті, а також невелика швидкодія процесорів (сучасні персональні комп'ютери ма-
ють швидкодію на три-чотири порядки більшу ніж суперкомп'ютери 60-тих років 20-го століття);
2) незручність, складність і повільність процедури взаємодії користувача з комп'ютером;
3) доступ до ресурсів комп'ютера мало дуже обмежене коло професіоналів.

Історія створення комп`ютерних мереж починається з кінця 50-х років 20-го сторіччя. У 1957 році за ініціативою Міністерства оборони США Агентство передових оборонних дослідницьких проектів США (DARPA) запропонувало проект розробки першої комп'ютерної мережі. Розробка мережі була доручена Каліфорнійському університету в Лос-Анджелесі, Стенфордському дослідницькому центру, Університету Юти та Університету штату Каліфорнія в Санта-Барбарі. Комп'ютерна мережа була названа ARPANET (Advanced Research Projects Agency Network), і в кінці 1969 року відбувся перший успішний сеанс зв'язку.

Паралельно з розвитком термінальних систем і комп'ютерних мереж стрімкими темпами йшов розвиток апаратно-технологічної бази: так тільки з 1990 року ємності жорстких дисків зросли в 10000 разів, швидкодія процесора збільшилась більше ніж у 250 разів, при цьому вартість цих пристроїв зменшилася майже в 10 разів. З'явились принципово нові пристрої - CD, DVD, диски blue-ray, флеш-пам'ять, безпровідні засоби міжкомп'ютерних зв'язків тощо.

Поява персональних комп'ютерів послужила стимулом для подальшого розвитку комп'ютерних мереж. Вони були достатньо дешевими та ідеальними елементами для побудови таких мереж. Розвитку комп'ютерних мереж сприяла розробка стандартних технологій об'єднання комп'ютерів у мережі: Ethernet, Arcnet, Token Ring, FDDI.

Нові технологічні розробки, в першу чергу в галузі комп'ютерних мереж, зробили можливою передачу комп'ютерними мережами голосу, відео зображення, малюнків у реальному часі – в режимі on-line.

Всі обчислювальні мережі можна класифікувати за рядом ознак. ***Залежно від відстаней між персональними комп'ютерами*** розрізняють такі обчислювальні мережі:

- локальні обчислювальні мережі - ЛОМ (LAN - Local Area Networks) - комп'ютерні мережі, розташовані в межах невеликої обмеженої території (будівлі чи в сусідніх будівлях) не більше як за 10 - 15 км.;

- територіальні обчислювальні мережі, які охоплюють значний географічний простір. До територіальних мереж можна віднести мережі регіональні (MAN - Metropolitan Area Network) і глобальні (WAN - Wide Area Network), тобто такі, що мають регіональні чи глобальні масштаби відповідно. Регіональні мережі зв'язують

абонентів району, міста чи області. Глобальні мережі об'єднують абонентів, які віддалені між собою на значну відстань, знаходяться в різних країнах чи континентах;

На сьогодні на підприємствах і в установах знайшли широке застосування локальні обчислювальні мережі, основне призначення яких забезпечити швидкий і зручний доступ до розподілених ресурсів (загальних, тобто таких, що спільно використовуються). Крім того, ЛОМ дозволяють проводити селекторні наради (відеоконференції), керівництву швидко і в наочній формі доводити до персоналу адміністративну, нормативнуі різноманітну поточну інформацію, співробітникам підприємства оперативно обмінюватися між собою інформацією тощо.

Локальні обчислювальні мережі забезпечують:

- розподіл даних. Дані в ЛОМ зберігаються на центральному комп'ютері і можуть бути доступними на робочих станціях, тому на кожному робочому місці не треба мати накопичувачів для зберігання однієї і тієї ж інформації;

- розподіл інформаційних і технічних ресурсів, тобто логічних дисків та інших зовнішніх запам'ятовуючих пристроїв (накопичувачі на CD-ROM, DVD, ZIP тощо); розподіл каталогів (тек) і файлів, що містяться в них; розподіл підключених до комп'ютера пристроїв: принтерів, модемів та інших зовнішніх пристроїв (дозволяє економно використовувати ресурси, наприклад, друкуючі пристрої, модеми тощо);

- розподіл програм. Всі користувачі локальних обчислювальних мереж можуть спільно мати доступ до програм (мережевих версій), які централізовано встановлюються в мережі;

- оперативний обмін інформацією в режимах on-line і off-line за допомогою intranet-сервісів.

В літературі розглядаються і інші класифікації комп'ютерних мереж. Класифікуючих ознак можна виділити чимало, але з практичної точки зору основними є такі:

- за рівнем управління;
- за типом використовуваних комп'ютерів;
- за адміністративними відносинами між комп'ютерами.

За типом використовуваних комп'ютерів можна виділити:

- однорідні мережі, які містять однотипні комп'ютери і системне програмне забезпечення;
- неоднорідні мережі, які містять різнотипні комп'ютери і різне системне програмне забезпечення.

За адміністративними відносинами між комп'ютерами можна виділити:

- ЛОМ із централізованим управлінням (з виділеними серверами);
- ЛОМ без централізованого управління (децентралізовані) або однорангові (однорівневі) мережі.

У локальних мережах із централізованим управлінням один із комп'ютерів є сервером, а інші - робочими станціями.

Сервери - це високопродуктивні комп'ютери з вінчестерами великої ємності і з високошвидкісною мережевою картою, які відповідають за зберігання даних, організацію доступу до цих даних і передачу даних робочим станціям або клієнтам. Комп'ютери, з яких здійснюється доступ до інформації на сервері, називаються робочими станціями або клієнтами.

У мережах із децентралізованим управлінням немає єдиного центру управління взаємодією робочих станцій і єдиного комп'ютера для зберігання даних. Однорангова локальна мережа - це ЛОМ рівноправних комп'ютерів,
кожний із яких має унікальне ім'я і, як правило, пароль для входу в нього у момент завантаження операційної системи.

Рівноправність комп'ютерів означає, що адміністратор кожного комп'ютера в локальній мережі може перетворити свій локальний ресурс у той, що розподіляється, встановлювати права доступу до нього і паролі. Він же відповідає за збереження або працездатність цього ресурсу. Локальний ресурс – це ресурс, доступний тільки з комп'ютера, на якому він знаходиться. Його ресурс, доступний для інших комп'ютерів, називається таким, що розподіляється або спільно використовується.

Таким чином, однорангова локальна мережа - це ЛОМ, в якій кожна робоча станція може розподілити всі або деякі свої ресурси з іншими робочими станціями мережі. Але відсутність виділеного сервера не дозволяє адміністраторові централізовано управляти всіма ресурсами однорангової локальної мережі.

Кожна робоча станція може виконувати функції, як клієнта, так і сервера, тобто надавати ресурси іншим робочим станціям і використовувати ресурси інших робочих станцій.

Однорангові локальні мережі можуть бути організованими на базі всіх сучасних операційних систем персональних комп'ютерів. До переваг однорангової локальної мережі відносяться:

- низька вартість;
- простота встановлення;
- висока надійність.

Така організація мережі дозволяє зберігати її працездатність практично при будь-якій кількості доступних вузлів.

До недоліків таких ЛОМ варто віднести:

- слабкий захист інформації;
- складність поновлення і зміни програмного забезпечення робочих станцій.

У локальних мережах із централізованим управлінням сервер забезпечує взаємодії між робочими станціями, виконує функції зберігання даних загального користування, організовує доступ до цих даних і

передає дані клієнтові. Клієнт опрацьовує одержані дані і надає результати опрацювання користувачеві. Необхідно відзначити, що опрацювання даних може здійснюватися і на серверу.

Локальні мережі з централізованим управлінням, в яких сервер призначений тільки для зберігання і видачі клієнтам інформації за запитами, називаються мережами з виділеним файл-сервером. Системи, в яких на сервері разом із зберіганням здійснюється і опрацювання інформації, називаються системами з архітектурою (або технологією) "клієнт-сервер".

Необхідно відзначити, що в серверних локальних мережах клієнтові безпосередньо доступні тільки ресурси серверів. Але робочі станції, що входять у ЛОМ із централізованим управлінням, можуть одночасно організувати між собою однорангову локальну мережу зі всіма її можливостями.

Враховуючи різноманіття послуг, що надаються комп'ютерними мережами, існує кілька типів серверів, а саме: мережевий сервер, файловий сервер, сервер друкування, поштовий сервер та ін.

Мережевий сервер є спеціалізованим комп'ютером, орієнтованим на виконання основного обсягу обчислювальних робіт і функцій із управління комп'ютерною мережею. Цей сервер містить ядро мережевої операційної системи, під управлінням якої здійснюється робота всієї локальної мережі. Мережевий сервер має досить високу швидкодію і великий обсяг пам'яті. При подібній мережевій організації функції робочих станцій зводяться до введення-виведення інформації та обміну нею з мережевим сервером.

Термін *файловий сервер* відноситься до комп'ютера, основною функцією якого є зберігання, управління і передача файлів. Він не обробляє і не змінює файли. Сервер може "не знати", чи є файл текстовим документом, графічним зображенням чи електронною таблицею. У загальному випадку на файловому сервері може навіть бути відсутньою клавіатура і монітор. Всі зміни у файлах даних здійснюються з клієнтських робочих станцій. Для цього клієнти зчитують файли із файлового сервера, здійснюють необхідні зміни даних і повертають їх назад на файловий сервер. Подібна організація найбільш ефективна при роботі великої кількості користувачів із загальною базою даних. У рамках великих мереж може одночасно використовуватися кілька файлових серверів.

Сервер друкування (*принт-сервер*) є друкуючим пристроєм, який за допомогою мережевого адаптера підключається до передавального середовища. Подібний мережевий друкувальний пристрій є самостійним і працює незалежно від інших мережевих пристроїв. Сервер друкування обслуговує заявки на друкування від усіх серверів і робочих станцій. За сервери друкування використовуються спеціальні високопродуктивні принтери.

Сервер застосувань - це потужний комп'ютер, що працює в мережі, має прикладну програму, з якою можуть працювати клієнти; застосування за запитами користувачів виконуються безпосередньо на серверу, а на робочу станцію передаються лише результати запиту.

Поштовий сервер використовується для організації обміну електронною кореспонденцією і зберіганням вхідних і вихідних повідомлень у електронних поштових скриньках;

Проксі-сервер - це ефективний засіб ввімкнення локальних мереж до мережі Internet; проксі-сервер – це комп'ютер, постійно ввімкнений до мережі Internet, через який відбувається спілкування користувачів локальної мережі з мережею Internet і укомплектований потужною системою захисту інформації.

Для ефективної взаємодії з мережею Internet можуть використовуватися *Web-сервери*.

До переваг ЛОМ із централізованим управлінням відносяться:

- вища швидкість опрацювання даних;
- ці мережі мають надійну систему захисту інформації і забезпечення секретності;
- простіші в управлінні порівняно з одноранговими мережами.

До недоліків ЛОМ із централізованим управлінням відносяться:

- мережа дорожча із-за виділеного сервера;
- менш гнучка порівняно з рівноправною мережею;
- надійність функціонування всієї мережі цілком залежить від працездатності сервера.

Компоненти комп’ютерних мереж

Будь-яка комп’ютерна мережа в загальному випадку включає такі **компоненти:**

1. Інформаційне забезпечення.
2. Лінгвістичне забезпечення.
3. Математичне забезпечення.
4. Програмне забезпечення.
5. Технічне забезпечення.
6. Нормативно-правове забезпечення.
7. Організаційне забезпечення.

Інформаційне забезпечення **–** це сукупність методів і засобів розміщення, подання та організації інформації, а також масивів даних, що зберігаються і циркулюють у мережі.

Лінгвістичне забезпечення є сукупністю мовних засобів, які використовуються на різних стадіях створення та експлуатації комп’ютерної мережі для підвищення ефективності розробки і забезпечення зручності спілкування абонента з ЕОМ та мережею.

Математичне забезпечення охоплює сукупність математичних моделей, методів та алгоритмів, які використовуються в більшості випадків на етапі розробки комп’ютерної мережі.

Програмне забезпечення – це сукупність програмних засобів для створення та експлуатації комп'ютерної мережі засобами обчислювальної техніки.

Технічне забезпечення є комплексом технічних засобів, які забезпечують функціонування комп'ютерної мережі і включають в себе пристрої, що реалізують типові операції опрацювання даних як поза ЕОМ (периферійні технічні засоби збирання, реєстрації, початкового опрацювання інформації, оргтехніка різного призначення, засоби телекомунікації та зв'язку), так і в ЕОМ різних класів.

Нормативно-правове забезпечення є сукупністю правових норм, які регламентують створення та функціонування комп'ютерної мережі. Правове забезпечення розробки комп'ютерної мережі включає нормативні акти договірних взаємовідносин між замовниками та розробниками комп'ютерної мережі. Правове забезпечення функціонування комп'ютерної мережі включає умови надання юридичної сили документам, одержаним із застосуванням обчислювальної техніки; права, обов'язки та відповідальність персоналу, в тому числі за своєчасність і точність опрацювання інформації; правила користування інформацією і порядок розв'язання суперечок із приводу її достовірності та інше.

Під ***організаційним забезпеченням*** комп'ютерних мереж розуміється сукупність методів і засобів, які дозволяють вдосконалювати організаційну структуру об'єктів, та управлінських функцій, що виконуються структурними підрозділами; визначити штатний розклад і чисельний склад кожного структурного підрозділу; розробити посадові інструкції персоналу управління в умовах функціонування комп'ютерної мережі.

3 Технічне забезпечення ЛОМ

Основними складовими елементами технічного забезпечення ЛОМ є:

- комп'ютери,
- лінії зв'язку,
- комунікаційне обладнання.

Комп'ютерні мережі можуть об'єднувати різноманітні комп'ютери як за класом (від персональних до супер-ЕОМ), за типом (різних апаратних і програмних платформ), так і за призначенням – абонентські (робочі станції) і серверні ЕОМ.

Лінія зв'язку складається в загальному випадку з фізичного середовища, яким передаються електричні інформаційні сигнали (дані), апаратури приймання-передавання даних і проміжної апаратури. За фізичне середовище в комунікаціях використовуються: метали (в основному мідь), надпрозоре скло (кварц) або пластикі ефір.

Однією лінією зв'язку можна утворити декілька каналів зв'язку (віртуальних чи логічних), наприклад шляхом частотного чи часового розподілу каналів. Канал зв'язку - це засіб односторонньої передачі даних. Якщо лінія зв'язку монопольно використовується каналом зв'язку, то в цьому випадку лінія зв'язку називається каналом зв'язку.

Канали передачі даних - це засоби двостороннього обміну даними, які включають лінії зв'язку і апаратуру передачі (приймання) даних. Канали передачі даних зв'язують між собою джерела інформації і приймачі інформації.

Оскільки апаратура передачі і приймання даних працює з даними в дискретному вигляді (тобто одиницямі нулям даних відповідають дискретні електричні сигнали), то при їх передачі аналоговим каналом потрібне перетворення дискретних даних в аналогові (модуляція).

При прийманні аналогових даних необхідне зворотне перетворення - демодуляція. ***Модуляція/демодуляція*** - процеси перетворення цифрової інформації в аналогові сигнали і навпаки. При модуляції інформація подається синусоїдальним сигналом тієї частоти, яку добре передає канал передачі даних.

Існують наступні *способи модуляці*:

- *амплітудна* модуляція;
- *частотна* модуляція;
- *фазова* модуляція.

При передачі дискретних сигналів цифровим каналом передачі даних використовується *кодування*:

- *потенційне*;
- *імпульсне*.

Таким чином, потенційне або імпульсне кодування застосовується на каналах високої якості, а модуляція на основі синусоїдальних сигналів переважно в тих випадках, коли канал вносить сильні спотворення до сигналів, які передаються.

Як правило, модуляція використовується у мережах при передачі даних аналоговими телефонними каналами зв'язку, які були розроблені для передачі голосу в аналоговій формі і тому погано підходять для безпосередньої передачі імпульсів.

Залежно від способів синхронізації ***канали передачі даних*** обчислювальних мереж можна розділити на ***синхронні і асинхронні***. Синхронізація необхідна для того, аби передавальний вузол даних міг передати сигнал приймаючому вузлу про початок передачі даних.

Синхронна передача даних вимагає додаткової лінії зв'язку для передачі синхронізуючих імпульсів. Передача бітів передавальною станцією і їх приймання приймаючою станцією здійснюється в моменти появи синхроімпульсів. В сучасних мережах для синхронізації використовується спеціальне кодування (наприклад, манчестерський код), яке не вимагає додаткової лінії для синхроімпульсу.

При *асинхронній* передачі даних додаткової лінії зв'язку не потрібно. В цьому випадку передача даних здійснюється блоками фіксованої довжини (байтами). Синхронізація здійснюється додатковими бітами (старт-бітами і стоп-бітами), які передаються перед байтом, який передається, і після нього.

При обміні даними між вузлами обчислювальних мереж використовуються *три* ***методи передачі даних:***

- *симплексна* (односпрямована) передача;
- *напівдуплексна* (приймання/передача інформації здійснюється почергово);
- *дуплексна* (двоспрямована), при якій кожний вузол одночасно передає і приймає дані (наприклад, переговори телефоном).

Лінії зв'язку. Залежно від фізичного середовища передачі даних лінії зв'язку можна розділити на:

- ***дротяні*** лінії зв'язку без ізолюючих і екрануючих обплетень;
- ***кабельні,*** де для передачі сигналів використовуються кабелі "вита пара", коаксіальні кабелі або оптоволоконні кабелі;
- ***безпровідні*** (радіоканали наземного і супутникового зв'язку), такі, що використовують для передачі сигналів електромагнітні хвилі, які розповсюджуються ефіром.

Дротяні лінії зв'язку використовуються для передачі телефонних і телеграфних сигналів, а також для передачі комп'ютерних даних. Ці лінії застосовуються як магістральні лінії зв'язку. Дротяними лініями зв'язки можуть бути організовані аналогові і цифрові канали передачі даних. Швидкість передачі дротяними є досить низькою. Крім того, до недоліків цих ліній відносяться недостатня перешкодозахищеність і можливість несанкціонованого підключення до мережі.

Поява кабельних систем дала поштовх розвитку саме локальних обчислювальних мереж, оскільки кабель дозволяв досягти значно більшої швидкості передачі даних ніж існуючі на той час телефонні лінії зв'язку.

Кабельні лінії зв'язку. В ЛОМ використовуються три **типи кабелів.**

1. ***Кабель на основі кручених пар*** являє собою декілька пар скручених попарно ізольованих мідних проводів у єдиній діелектричній (пластиковій) оболонці. Він досить гнучкий і зручний для прокладки. Скручування проводів дозволяє звести до мінімуму індуктивні наведення кабелів один на одного й знизити вплив перехідних процесів.

Як правило, в кабель входить дві (рис. 1.1) або чотири кручені пари.

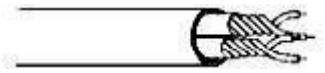

Рис. 3.1. Кабель із крученими парами.

Неекрановані кручені пари *UTP* характеризуються слабкою захищеністю від зовнішніх електромагнітних перешкод, а також від підслуховування, що може здійснюватися з метою, наприклад, промислового шпигунства. Причому перехоплення переданої мережею інформації можливе як за допомогою контактного методу (наприклад, за допомогою двох голок, уткнутих у кабель), так і за допомогою безконтактного методу, що зводиться до радіоперехоплення випромінюваних кабелем електромагнітних полів. Причому дія перешкод і величина випромінювання збільшується зі зростанням довжини кабелю. Для усунення цих недоліків застосовується екранування кабелів.

У випадку екранованої кручений пари STP кожна із кручених пар міститься в металевій обгортці-екрані для зменшення випромінювань кабелю, захисту від зовнішніх електромагнітних перешкод і зниження взаємного впливу пар проводів один на одного (перехресні наведення). Аби екран захищав від перешкод, він повинен бути заземленим. Звичайно, екранована кручена пара помітно дорожча, ніж неекранована. Її використання вимагає спеціальних екранованих роз'ємів. Тому зустрічається вона значно рідше, ніж неекранована кручена пари.

До основних переваг неекранованих кручених пар відносяться простота монтажу роз'ємів на кінцях кабелю і простота ремонту будь-яких ушкоджень порівняно з іншими типами кабелю**.** Всі інші характеристики цих кабелів гірші, ніж в інших.

2. Коаксіальний кабель являє собою електричний кабель, що складається із центрального мідного провідника (1) і металевого обплетення – екрана (3), розділених між собою прошарком діелектрика - внутрішньої ізоляції (2) і розміщених у загальну зовнішню оболонку (4) (рис. 1.2).

Рис. 3.2. Коаксіальний кабель.

Коаксіальний кабель донедавна був досить популярним, що пов'язано з його високою перешкодозахищеністю (завдяки металевому обплетенню), більш широкими, чим у крученої пари, смугами пропускання (понад 1ГГц), а також більшими допустимими відстанями передачі (до кілометра). До нього важче механічно підключитися для несанкціонованого прослуховування мережі, він дає помітно менше

електромагнітних випромінювань зовні. Проте монтаж і ремонт коаксіального кабелю істотно складніший, чим крученої пари, а вартість його вища (він дорожчий в 1,5 – 3 рази). Складніша і установка роз'ємів на кінцях кабелю.

Існує два основних типи коаксіального кабелю:

- тонкий кабель, що має діаметр близько 5 мм;
- товстий кабель, діаметром близько 10 мм. Він являє собою класичний варіант коаксіального кабелю і на сьогодні майже повністю витіснений тонким кабелем.

Тонкий кабель використовується для передачі на менші відстані, чим товстий, оскільки сигнал у ньому загасає сильніше. Зате з тонким кабелем набагато зручніше працювати: його можна оперативно прокласти до кожного комп'ютера, а товстий вимагає твердої фіксації на стіні приміщення.

Підключення до тонкого кабелю простіше й не вимагає додаткового устаткування. А для підключення до товстого кабелю треба використовувати спеціальні досить дорогі пристрої, що проколюють його оболонки і установлюють контакт як із центральною жилою, так і з екраном. Товстий кабель приблизно вдвічі дорожчий, ніж тонкий, тому тонкий кабель застосовується набагато частіше.

Існують варіанти коаксіального кабелю з подвійним екраном (один екран розташований всередині іншого й відділений від нього додатковим прошарком ізоляції). Такі кабелі мають кращу перешкодозахищеністьі захист від прослуховування, але вони небагато дорожчі звичайних.

На сьогодні вважається, що коаксіальний кабель застарів, у більшості випадків його може замінити кручена пара або оптоволоконний кабель. І нові стандарти на кабельні системи вже не включають його в перелік типів кабелів.

3. ***Оптоволоконний кабель*** – це принципово інший тип кабелю порівняно з розглянутими двома типами кабелю. Інформація ним передається не електричним сигналом, а світловим. Головний його елемент - прозоре скловолокно, яким світло проходить на величезні відстані (до десятків кілометрів) з незначним ослабленням.

Структура оптоволоконного кабелю досить проста й схожа на структуру коаксіального кабелю (рис. 1.3). Тільки замість центрального мідного проводу тут використовується тонке (діаметром близько 1 - 10 напівтемних) скловолокно (3), а замість внутрішньої ізоляції - скляна або пластикова оболонка (2), яка не дозволяє світлу виходити за межі скловолокна. Металеве обплетення кабелю відсутнє, оскільки екранування від зовнішніх електромагнітних перешкод тут не потрібно. Проте іноді його застосовують для механічного захисту від навколишнього середовища (такий кабель іноді називають броньовим, він може поєднувати під одною оболонкою декілька оптоволоконних кабелів).

Оптоволоконний кабель має виняткові характеристики з перешкодостійкості і таємності переданої інформації. Ніякі зовнішні електромагнітні перешкоди не здатні спотворити світловий сигнал, а сам сигнал не породжує зовнішніх електромагнітних випромінювань. Підключитися до цього типу кабелю для несанкціонованого прослуховування мережі практично неможливо, оскільки при цьому порушується цілісність кабелю. Теоретично можлива смуга пропускання такого кабелю досягає величини 10^{12} Гц, тобто 1000 ГГц, що незрівнянно вище, ніж в електричних кабелів. Вартість оптоволоконного кабелю постійно знижується й зараз приблизно дорівнює вартості тонкого коаксіального кабелю.

Типова величина загасання сигналу в оптоволоконних кабелях на частотах, що використовуються у локальних мережах, становить від 5 до 20 дБ/км, що приблизно відповідає показникам електричних кабелів на низьких частотах. У випадку оптоволоконного кабелю при зростанні частоти переданого сигналу загасання збільшується несуттєво і на більших частотах (особливо понад 200 МГц) його переваги перед електричним кабелем незаперечні.

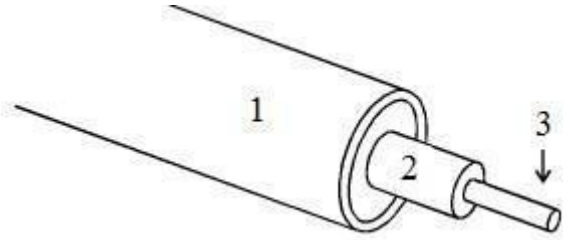

Рис. 3.3. Структура оптоволоконного кабелю.

Проте оптоволоконний кабель має деякі недоліки. Головний із них – це висока складність монтажу (при установці роз'ємів необхідна мікронна точність, від точності відколу скловолокна й міри його полірування залежить загасання в роз'ємі). Для установки роз'ємів застосовуються зварювання або склеювання за допомогою спеціального гелю, що має такий же коефіцієнт переломлення світла, що й скловолокно. Тут потрібна висока кваліфікація персоналу й спеціальні інструменти.

Використання оптоволоконного кабелю вимагає спеціальних оптичних приймачів і передавачів, що перетворюють світлові сигнали в електричні і навпаки, що часом істотно збільшує вартість мережі в цілому.

Оптоволоконний кабель менш міцний і гнучкий, чим електричний. Типова величина допустимого радіуса вигину становить близько 10 - 20 см, при менших радіусах вигину центральне волокно може зламатися. Погано переносить кабель і механічне розтягання, а також роздавлюючі впливи.

Чутливий оптоволоконний кабель і до іонізуючих випромінювань, із-за яких знижується прозорість скловолокна, тобто збільшується загасання сигналу. Різкі перепади температури також негативно позначаються на ньому, скловолокно може тріснути.

Застосовують оптоволоконний кабель тільки в мережах з топологією зірка й кільце. Ніяких проблем узгодження й заземлення в цьому випадку не існує.

4. Мережеві топології

Всі комп'ютери в локальній мережі сполучені лініями зв'язку. Порядок з'єднання комп'ютерів називають топологією. При цьому розрізняють фізичну і логічну топологію. Логічна і фізична топології мережі незалежні одна від одної. Фізична топологія - це геометрія побудови мережі, тобто порядок з'єднання комп'ютерів лініями зв'язку, а логічна топологія визначає структуру зв'язків, характер поширення сигналів мережею. Якщо не уточнюють, то, як правило, мають на увазі саме логічну топологію.

Топологія визначає:

- вимоги до устаткування, необхідну складність мережевої апаратури,
- можливі типи середовищ передачі (каналів зв'язку), тип використовуваного кабелю,
- можливі й найбільш зручні методи управління обміном,
- міру відмовостійкості мережі,
- допустимий розмір мережі (довжина ліній зв'язку),
- максимальну кількість абонентів,
- необхідність електричного узгодження,
- можливості розширення мережі і багато чого іншого.

Існує три базових топології комп'ютерних мереж:

- **«шина»** (bus)**,**
- **«зірка» (star),**
- **«кільце» (ring).**

В **топології «шина»** всі комп'ютери паралельно підключаються до однієї лінії зв'язку, інформація від кожного комп'ютера одночасно передається всім іншим комп'ютерам, але приймається тільки тим вузлом, якому вона призначається (рис. 3.5). «Шина» своєю структурою допускає ідентичність мережевого устаткування комп'ютерів, а також рівноправність всіх абонентів. При такому з'єднанні комп'ютери можуть передавати тільки почережно, оскільки лінія зв'язку єдина. У противному випадку передана інформація буде спотворюватися в результаті накладення (конфлікту, колізії). Таким чином, у шині реалізується режим напівдуплексного (half duplex) обміну (в обох напрямках, але почережно, не одночасно).

Переваги мереж із шинною топологією такі:

- відмова одного з вузлів не впливає на роботу мережі в цілому,
- мережу легко налаштовувати і конфігурувати,
- простота додавання нових абонентів,
- мінімальна кількість використовуваного кабелю,
- невисока вартість мережевого устаткування.

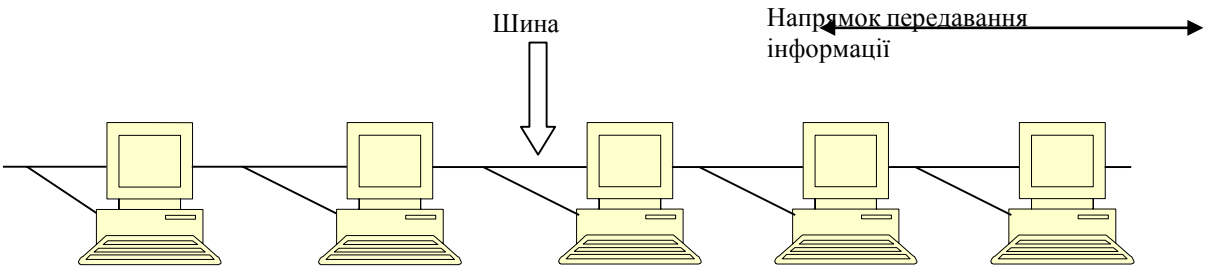

Рис. 3.5. Топологія «шина».

Недоліки мереж шинної топології такі:

- розрив кабелю або коротке замикання в будь-якій точці кабелю шини виводить із ладу всю мережу,
- будь-яку відмову мережевого устаткування в шині досить важко локалізувати, оскільки всі адаптери включені паралельно, і зрозуміти, який із них вийшов із ладу, не так-то просто,
- обмежена довжина кабелю і кількість робочих станцій,
- збільшення кількості робочих станцій зменшує реальну пропускну спроможність шини.

Шина широко використовувалась у перших класичних мережевих технологіях ArcNet і EtherNet. У

сучасних ЛОМ шина практично не використовується внаслідок перелічених недоліків.

У мережі, побудованою за топологією типу **“зірка”** (рис. 3.6), кожна робоча станція приєднується кабелем до концентратора або комутатора, який є центральним елементом зірки. Він забезпечує паралельне з'єднання з комп’ютером, або з іншим центральним елементом, таким чином, всі комп'ютери, підключені до мережі, можуть спілкуватися один із одним.

Переваги топології «зірка» такі:

- легко підключити новий периферійний елемент,
- фактично відсутнє згасання сигналів,
- мережа стійка до несправностей окремих робочих станцій і до розривів з'єднання з будь-якою з них,

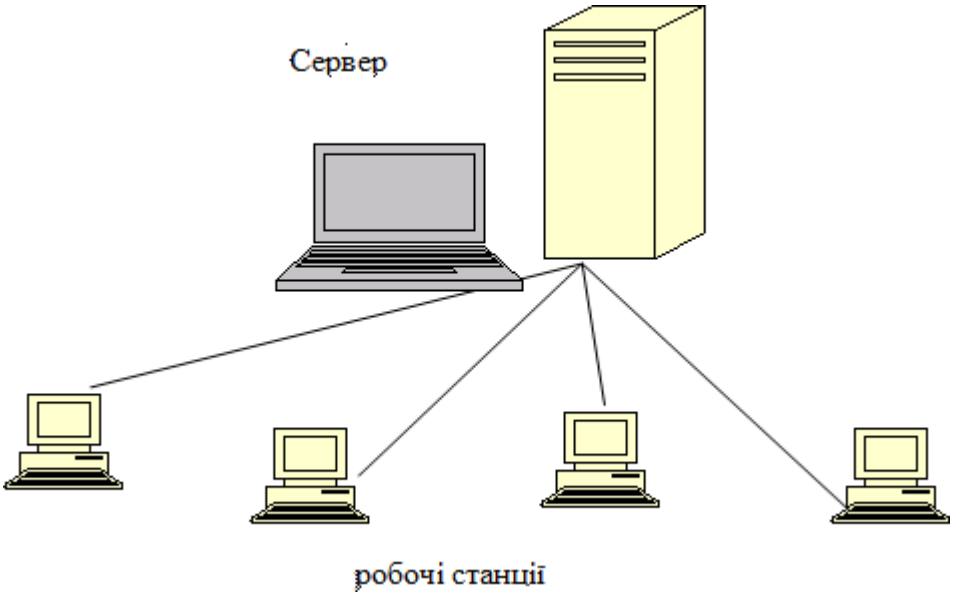

Рис. 3.6. Топологія «зірка».

- велика перевага зірки полягає в тому, що всі точки підключення зібрані в одному місці. Це дозволяє легко контролювати роботу мережі, локалізувати несправності шляхом простого відключення від центра тих чиінших абонентів (що неможливо, наприклад, у випадку шини), а також обмежувати доступ сторонніх осіб до життєво важливих для мережі точок підключення.

Недоліки топології «зірка» такі:

- будь-яка відмова центрального елемента робить мережу повністю непрацездатною,
- жорстке обмеження кількості абонентів, обумовлене кількістю вхідних роз’ємів центрального елемента,
- великі витрати кабелю.

У топології **«кільце»** кожний комп’ютер з’єднаний лініями зв’язку тільки із двома іншими: від одного він тільки одержує інформацію, а іншому тільки передає (рис. 3.7). Дані в кільці рухаються в одному і тому ж напрямі. На кожній лінії зв’язку, як і у випадку зірки, працює тільки один передавач і один приймач. Це дозволяє відмовитися від застосування зовнішніх терміна-
торів. Важлива особливість кільця полягає в тому, що кожний комп’ютер ретранслює (відновлює) сигнал, тобто виступає в ролі репітера, тому загасання сигналу у всьому кільці не має ніякого значення, важливе тільки загасання між сусідніми комп’ютерами кільця. Чітко виділеного центра в цьому випадку немає, всі комп’ютери можуть бути однаковими. Проте досить часто в кільці виділяється спеціальний абонент, що управляє обміном або контролює обмін.

Приймаюча робоча станція розпізнає і одержує тільки адресоване їй повідомлення.

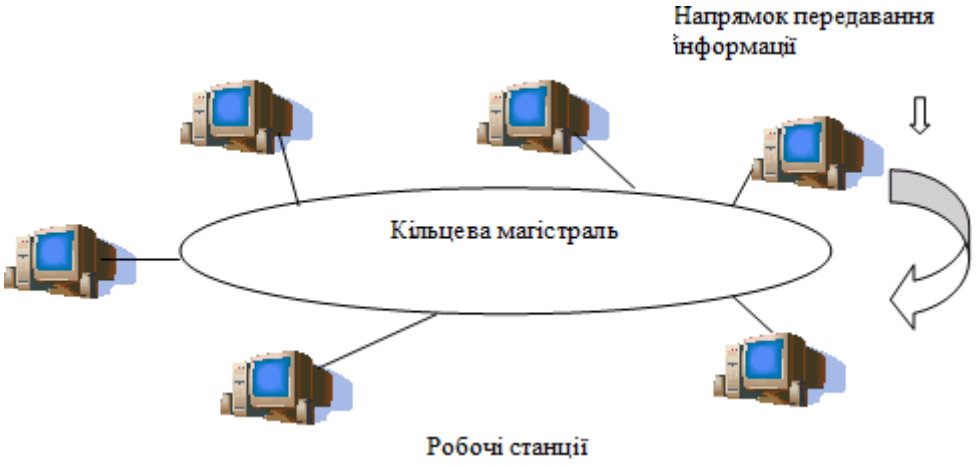

Рис. 3.7. Фізичне кільце

Переваги топології «кільце»:

- підключення нових абонентів вимагає обов'язкової зупинки роботи всієї мережі на час підключення,
- як і у випадку топології «шина», максимальна кількість абонентів у кільці може бути досить велика (до тисячі й більше),
- кільцева топологія є стійкою до перевантажень, вона забезпечує впевнену роботу із великими потоками інформації, оскільки в ній, як правило, немає конфліктів (на відміну від шини), а також відсутній центральний абонент (на відміну від зірки),
- ретрансляція сигналів кожним абонентом дозволяє істотно збільшити розміри всієї мережі в цілому (іноді до декількох десятків кілометрів); кільце щодо цього істотно перевершує будь-яку іншу топологію,
- мережу з топологією «кільце» досить легко створювати і налаштовувати.

Основним недоліком мереж із топологією «кільце» є те, що пошкодження лінії зв'язку в одному місці або відмова будь-якої робочої станції призводить до непрацездатності всієї мережі.

Як правило, в чистому вигляді топологія "кільце" не застосовується із-за своєї ненадійності, тому на практиці застосовуються різні модифікації кільцевої топології.

Іноді топологія «кільце» виконується на основі двох кільцевих ліній зв'язку, що передають інформацію в протилежних напрямках. Мета подібного рішення – збільшення (в ідеалі вдвічі) швидкості передачі інформації. До того ж при ушкодженні одного з кабелів мережа може працювати з іншим кабелем (правда, гранична швидкість зменшиться).

«Комірчаста топологія» комп'ютерних мереж існує скоріше у вигляді теоретичної концепції, ніж у вигляді практичної реалізації. У мережі з комірчастою топологією всі комп'ютери пов'язані один із одним окремими сполуками (рис. 3.8). У реальності ця топологія реалізована поки-що тільки в мережах із двома вузлами. При збільшенні кількості комп'ютерів у мережі кожний із них довелося б обладнати мереженими інтерфейсами за числом інших комп'ютерів. З іншого боку, мережа з комірчастою топологією має бездоганну відмовостійкість: будь-яка несправність в ній позначається на працездатності тільки одного комп'ютера.

У глобальних мережах комірчаста топологія використовується. В комірчастій мережі завдяки використанню надлишкових маршрутизаторів дані можуть дістатися від однієї системи до іншої декількома шляхами (рис. 3.9).

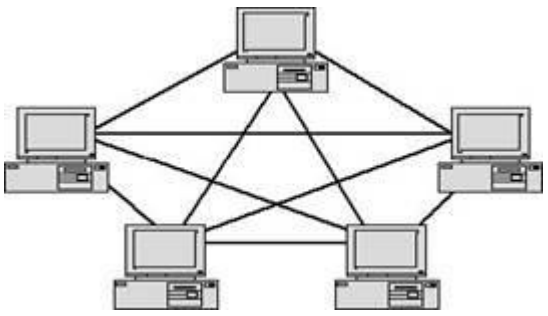

Рис. 3.8. Варіанти з'єднання комп'ютерів у мережі з «комірчастою» топологією

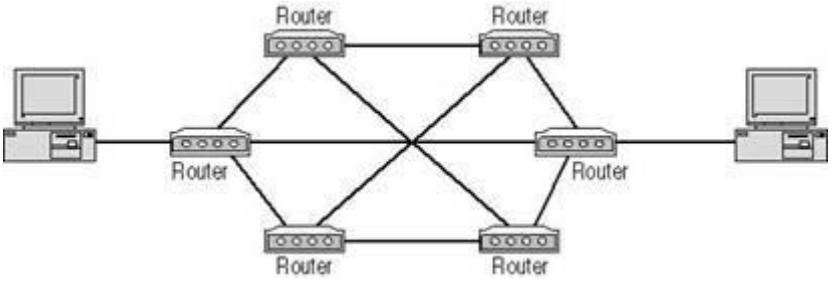

Рис. 3.9. Інтермережа з комірчастою топологією

Ця топологія часто застосовується у великих корпоративних мережах, оскільки вона захищає їх від несправностей маршрутизаторів (Router), концентраторів, кабелів та ін. Найчастіше, коли говорять про комірчасту топологію, мають на увазі саме таке її застосування.

Крім розглянутих базових топологій нерідко застосовуються також інші топології, які є різноманітними комбінаціями базових.

Теми доповідей

1. Побудова й використання комп'ютерних моделей.
2. Телекомунікації, телекомунікаційні мережі різного типу, їхнє призначення й можливості

Література

1. Зацеркляний М.М. Компютерні основи систем кібербезпеки: навч. посібник/ Зацеркляний М.М., Струков В.М.-Харків: Тов. «В деле», 2017.- 292 с.
2. Тарарака В.Д. Архітектура комп'ютерних систем: навчальний посібник. – Житомир : ЖДТУ, 2018. – 383 с.
3. Хіхловська І.В. Обчислювальна техніка та мікропроцесори : підручник / Хіхловська І.В., Антонов О.С. – Одеса: ОНАЗ ім. О.С. Попова, 2011. – 440 с

Тема № 4. Програмне забезпечення

Семінарське заняття: Програмне забезпечення ЕОМ

Мета заняття: Ознайомитись з програмним забезпеченням ЕОМ

Тривалість: 2 год

Навчальні питання

1. Програмне забезпечення
2. Операційні системи
3. Управління даними
4. Механізми захисту операційних систем
5. Аналіз захищеності сучасних операційних систем

1. Програмне забезпечення

В основу роботи комп'ютерів покладено програмний принцип управління, який полягає в тому, що комп'ютер виконує дії за заздалегідь заданою програмою. Цей принцип забезпечує універсальність використання комп'ютера: у певний момент часу розв'язується задача відповідно до вибраної програми. Після її завершення в пам'ять завантажується інша програма і т.д.

Програма - це запис алгоритму розв'язування задачі у вигляді послідовності команд або операторів мовою, яку розуміє комп'ютер. Кінцевою метою будь-якої комп'ютерної програми є управління апаратними засобами.

Для нормального розв'язування задач на комп'ютері потрібно, аби програма була налагоджена, не вимагала доопрацювань і мала відповідну документацію. Тому, щодо роботи на комп'ютері часто використовують термін програмне забезпечення (software), під яким розуміється сукупність програм, процедур, правил, а також документації, що стосуються функціонування системи обробки даних.

Програмне та апаратне забезпечення в комп'ютері працюють у нерозривному зв'язку і взаємодії. Склад програмного забезпечення обчислювальної системи називається програмною конфігурацією. Між програмами існує взаємозв'язок, тобто робота множини програм базується на програмах нижчого рівня.

Міжпрограмний інтерфейс - це розподіл програмного забезпечення на декілька пов'язаних між собою рівнів. Рівні програмного забезпечення є пірамідою, де кожний вищий рівень базується на програмному забезпеченні попередніх рівнів. Схематично структура програмного забезпечення наведена на рис. 4.1.

прикладний рівень

службовий рівень
системний рівень
базовий рівень

Рис. 4.1 - Рівні програмного забезпечення

Базовий рівень є нижчим рівнем програмного забезпечення. Він відповідає за взаємодію з базовими апаратними засобами. Базове програмне забезпечення міститься у складі базового апаратного забезпечення і зберігається у спеціальних мікросхемах постійної пам'яті (ПЗП), утворюючи базову систему введення- виведення BIOS. Програми та дані записуються в ПЗП на етапі виробництва і не можуть бути змінені під час експлуатації.

Системний рівень - є перехідним. Програми цього рівня забезпечують взаємодію інших програм комп'ютера з програмами базового рівня і безпосередньо з апаратним забезпеченням. Від програм цього рівня залежать експлуатаційні показники всієї обчислювальної системи. При під'єднанні до комп'ютера нового обладнання на системному рівні повинна бути встановлена програма, яка забезпечує для решти програм взаємозв'язок із пристроєм. Конкретні програми, призначені для взаємодії з конкретними пристроями, називаються драйверами.

Інший клас програм системного рівня відповідає за взаємодію з користувачем. Завдяки йому, можна вводити дані в обчислювальну систему, управляти її роботою й одержувати результат у зручній формі. Це засоби забезпечення користувацького інтерфейсу, від них залежить зручність та продуктивність роботи з комп'ютером.

Сукупність програмного забезпечення системного рівня утворює ядро операційної системи комп'ютера. Наявність ядра операційної системи - це перша умова для можливості практичної роботи користувача з обчислювальною системою. Ядро операційної системи виконує такі функції: управління пам'яттю, процесами введення-виведення, файловою системою, організацією взаємодії та диспетчеризації процесів, облік використання ресурсів, опрацювання команд тощо.

Програми службового рівня взаємодіють як із програмами базового рівня, так і з програмами системного рівня. Призначення службових програм (утиліт) полягає в автоматизації робіт із перевірки та налаштування комп'ютерної системи, а також для покращення функцій системних програм. Деякі службові програми (програми обслуговування) відразу входять до складу операційної системи, доповнюючи її ядро, але більшість єзовнішніми програмами і розширюють функції операційної системи. Тобто, в розробці службових програм відслідковуються два напрямки: інтеграція з операційною системою та автономне функціонування.

2. Операційні системи

Операційна система (ОС, operating system, OS) – це комплекс управляючих і опрацьовуючих програм, які, з одного боку, виступають як інтерфейс між пристроями обчислювальної системи і прикладними програмами, а з іншого боку - призначені для управління пристроями, управління обчислювальними процесами, управління ефективним розподілом обчислювальних ресурсів між обчислювальними процесами і організації надійних обчислень. Це визначення застосовне до більшості сучасних операційних систем загального призначення.

Є застосування обчислювальної техніки, для яких операційні системи не потрібні. Наприклад, вбудовані мікрокомп'ютери, що містяться у багатьох побутових приладах, автомобілях (іноді до десятка в кожному), найпростіших стільникових телефонах, постійно виконують лише одну програму, яка завантажується при ввімкненні. Чимало простих ігрових приставок також є спеціалізованими мікрокомп'ютерами і можуть обходитися без операційної системи, завантажуючи при ввімкненні програму, записану на вставленому в пристрій «картриджі» або компакт-диску.

Операційні системи потрібні, якщо:

• обчислювальна система використовується для різних задач, причому програми, які розв’язують ці задачі, потребують збереження даних та обміну ними. З цього випливає необхідність універсального механізму збереження даних; у переважній більшості випадків в операційних систем ця необхідність реалізується файловою системою. Сучасні операційні системи, крім того, надають можливість безпосередньо «зв'язати» результат однієї програми із введенням іншої, минаючи відносно повільні дискові операції;

• різні програми потребують виконання одних і тих же рутинних дій. Наприклад, просте введення символу з клавіатури і відображення його на екрані може вимагати виконання сотень машинних команд, а дискова операція - тисяч. Аби не програмувати їх щоразу заново, операційні системи надають системні бібліотеки часто використовуваних підпрограм (функцій);

• між програмами та користувачами системи необхідно розподіляти повноваження, аби користувачі могли захищати свої дані від несанкціонованого доступу, а можлива помилка в програмі не викликала тотальних неприємностей;

• необхідна можливість імітації «одночасного» виконання декількох програм на одному комп'ютері (якщо він містить лише один процесор), що здійснюється за допомогою прийому, відомого як «розподіл часу». При цьому спеціальний компонент, який називається планувальником, ділить процесорний час на короткі відрізки і надає їх почережно різним виконуваним програмам (процесам);

• оператор повинен мати можливість так чи інакше управляти процесами виконання окремих програм. Для цього використовуються операційні середовища - оболонки і набори утиліт - вони можуть бути частиною

операційної системи.

Таким чином, сучасні універсальні операційні системи можна охарактеризувати, насамперед, як такі, що:

• використовують файлові системи (з універсальним механізмом доступу до даних),

• є багатокористувацькими (із розподілом повноважень),

• є багатозадачними (із розподілом часу).

У логічній структурі типової обчислювальної системи операційна система займає положення між пристроями з їх мікроархітектурою, машинною мовою, можливо, власними (вбудованими) мікропрограмами - з одного боку - і прикладними програмами з іншого.

Розробникам програмного забезпечення операційна система дозволяє абстрагуватися від деталей реалізації та функціонування пристроїв, надаючи мінімально необхідний набір функцій.

У більшості обчислювальних систем операційна система є основною, найбільш важливою (а іноді і єдиною) частиною системного програмного забезпечення. З 1990-х років найбільш поширеними операційними системами є системи ряду Windows і системи класу UNIX (особливо Linux і Mac OS).

При розгляді засад функціонування ОС прийнято виділяти чотири основні групи функцій, виконуваних системою:

• управління пристроями; маються на увазі всі периферійні пристрої, що вмикаються до комп'ютера, - клавіатура, монітор, принтери, диски і т.п.;

• управління даними; під цим терміном розуміється робота з файлами;

• управління процесами; ця сторона роботи ОС пов'язана із завантаженням і завершенням роботи програм, обробкою помилок, забезпеченням паралельної роботи декількох програм на одному комп'ютері;

• управління пам'яттю; оперативна пам'ять комп'ютера - це такий ресурс, якого завжди не вистачає; у цих умовах розумне планування використання пам'яті є найважливішим чинником ефективної роботи.

Є ще кілька важливих обов'язків, які лягають на ОС і які важко втиснути в рамки традиційної класифікації функцій. До них, насамперед, відносяться такі:

• організація інтерфейсу з користувачем; форми інтерфейсу можуть бути різноманітними, залежно від типуі призначення ОС: мова управління пакетами завдань, набір діалогових команд, засоби графічного інтерфейсу;

• захист даних; як тільки система перестає бути надбанням одного ізольованого від зовнішнього світу користувача, питання захисту даних від несанкціонованого доступу набувають першорядну важливість; ОС, яка забезпечує роботу в мережі або в системі розподілу часу, повинна відповідати наявним стандартам безпеки;

• ведення статистики; у ході роботи ОС повинна збиратися, зберігатися і аналізуватися різноманітна інформація: про кількість часу, витраченого різними програмами та користувачами, про інтенсивність використання ресурсів, про спроби некоректних дій користувачів, про збої устаткування і т.п.; зібрана інформація зберігається в системних журналах і в облікових записах користувачів.

Для розуміння роботи ОС необхідно вміти виділяти основні частини системи та їх зв'язки, тобто описувати структуру системи. Для різних ОС їх структурний поділ може бути досить різним. Найбільш загальними видами структуризації можна вважати два. З одного боку, можна вважати, що ОС розділена на підсистеми, відповідні перерахованим вище групам функцій. Такий поділ досить обґрунтований, програмні модулі ОС дійсно в основному можна віднести до однієї з цих підсистем. Інший важливий структурний поділ пов'язаний із поняттям ядра системи.

Ядро, як можна зрозуміти з назви, це основна, «найсистемніша» частина операційної системи. Є різні визначення ядра. Відповідно до одного з них, ядро - це резидентна частина системи, тобто до ядра належить той програмний код, який постійно знаходиться в пам'яті протягом всієї роботи системи. Інші модулі ОС є транзитними, тобто вони довантажуються в пам'ять із диска в міру необхідності на час своєї роботи. До транзитних частин системи відносяться :

• утиліти (utilities) - окремі системні програми, які вирішують окремі завдання, такі як форматування і перевірку дисків, пошук даних у файлах, моніторинг (відстеження) роботи системи та багато іншого;

• системні бібліотеки підпрограм, які дозволяють прикладним програмам використовувати різні спеціальні можливості, підтримувані системою (наприклад, бібліотеки для графічного виведення, для роботи з мультимедіа тощо);

• інтерпретатор команд - програма, що виконує введення команд користувача, їх аналіз і виклик з інших модулів для виконання команд;

• системний завантажувач - програма, яка при завантаженні ОС (наприклад, при ввімкненні живлення) забезпечує завантаження системи з диска, її ініціалізацію і старт;

• інші види програм у залежності від конкретної системи.

Як основоположний елемент операційної системи, ядро є найбільш низьким рівнем абстракції для доступу програм до ресурсів обчислювальної системи, необхідних для їх роботи. Як правило, ядро надає такий доступ виконуваним процесам відповідних застосувань за рахунок використання механізмів взаємодії між процесами та звертанню застосувань до системних викликів ОС.

Описана задача може різнитися залежно від типу архітектури ядра і способу її реалізації.

3. Управління даними

Файлова система - це частина операційної системи, призначення якої полягає в тому, аби забезпечити користувачеві зручний інтерфейс при роботі з даними, що зберігаються на диску, і забезпечити спільне

використання файлів кількома користувачами і процесами.

У широкому розумінні поняття "файлова система" включає:

• сукупність усіх файлів на диску;

• набори структур даних, що використовуються для управління файлами, такі, наприклад, як каталоги файлів, дескриптори файлів, таблиці розподілу вільного і зайнятого простору на диску;

• комплекс системних програмних засобів, які реалізують управління файлами, зокрема, створення, знищення, читання, запис, іменування, пошук та інші операції над файлами.

Файл – це з одного боку, формально не визначене поняття. З точки зору змісту, файл – це множина даних, об'єднаних деяким логічним зв'язком, тобто одна і та ж сукупність даних може розглядатися як один або декілька файлів (початкові дані до задачі; навчальні матеріали тощо). З іншого боку, поняття файл відноситься до об'єкту, який однозначно визначається такими ознаками:

- файл об'єднує множину даних;
- має ім'я;
- з метою довготривалого та надійного зберігання інформації розташовується на зовнішньому пристрої;
- передбачає багаторазове використання інформації з розривом у часі;
- передбачає спільне використання інформації декількома застосуваннями або користувачами одночасно (розподіляється ресурс) або з розривом у часі.

Прийняте означення є означенням файлу з точки зору користувача. Деталі розташування даних на зовнішньому пристрої і роботи з ними на низькому (фізичному) рівні файлова система бере на себе, екрануючи всі складнощі цього рівня і надаючи користувачеві зручну логічну модель і набір відповідних команд.

Користувачі дають файлам символьні імена, при цьому враховуються обмеження операційної системи як на використовувані символи, так і на довжину імені.

Файли бувають різних типів: звичайні файли, спеціальні файли, файли-каталоги.

Звичайні файли, у свою чергу, діляться на текстові та виконувані. Текстові файли складаються з рядків символів, поданих у ASCII-коді. Це -документи, початкові тексти програм тощо. Текстові файли можна прочитати на екрані і надрукувати на принтері. Двійкові файли не використовують ASCII-коди, вони часто мають складну внутрішню структуру. Всі операційні системи повинні вміти розпізнавати принаймні один тип файлів - їх власні виконувані файли.

Спеціальні файли - це файли, асоційовані з пристроями введення-виведення, які дозволяють користувачеві виконувати операції введення-виведення, використовуючи звичайні команди запису у файл або читання з файлу. Ці команди обробляються спочатку програмами файлової системи, а потім на деякому етапі виконання запиту перетворюються операційною системою в команди управління відповідним пристроєм. Спеціальні файли, так само як і пристрої введення-виведення, діляться на блок-орієнтовані і байт-орієнтовані.

Каталог - це, з одного боку, група файлів, об'єднаних користувачем виходячи з деяких міркувань, а з іншого боку - це файл, що містить системну інформацію про групу файлів, його складових. У каталозі міститься список файлів, що входять до нього, і встановлюється відповідність між файлами і їх характеристиками (атрибутами).

Механізми захисту операційних систем

Під механізмами захисту операційної системи розуміються всі засоби і механізми захисту даних, що функціонують у складі операційної системи. Операційні системи, у складі яких функціонують засоби і механізми захисту даних, часто називаються захищеними системами.

Під безпекою операційної системи розуміється такий її стан, при якому неможливе випадкове чи навмисне порушення функціонування операційної системи, а також порушення безпеки, при яких ресурси комп’ютерної системи знаходяться під управлінням операційної системи.

Вкажемо особливості операційної системи, які дозволяють виділити питання забезпечення її безпеки в особливу категорію:

- управління всіма ресурсами системи;
- наявність вбудованих механізмів, які прямо чи опосередковано впливають на безпеку програм і даних, що працюють у середовищі операційної системи;
- забезпечення інтерфейсу користувача з ресурсами системи;
- розміри і складність операційної системи.

Більшість операційних систем мають дефекти з погляду забезпечення безпеки даних у системі, що обумовлено виконанням завдання забезпечення максимальної доступності системи для користувача.

Розглянемо типові функціональні дефекти операційної системи, які можуть призвести до створення каналів витоку даних.

1. Ідентифікація. Кожному ресурсу в системі має бути присвоєно унікальне ім'я - ідентифікатор. У багатьох системах користувачі не мають можливості упевнитися в тому, що використовувані ними ресурси дійсно належать системі.

2. Паролі. Більшість користувачів вибирають найпростіші паролі, які легко підібрати або вгадати.

3. Список паролів. Зберігання списку паролів у незашифрованому вигляді дає можливість його компрометації з подальшим несанкціонованим доступом до даних.

4. Граничні значення. Для запобігання спроб несанкціонованого входу в систему за допомогою підбору

пароля необхідно обмежити число таких спроб, що в деяких операційних системах не передбачено.

5. Довіра. У багатьох випадках програми операційної системи вважають, що інші програми працюють правильно.

6. Спільна пам'ять. При використанні спільної пам'яті не завжди після виконання програм очищаються ділянки оперативної пам'яті (ОП).

7. Розрив зв'язку. У разі розриву зв'язку операційна система повинна негайно закінчити сеанс роботи з користувачем або повторно встановити справжність суб'єкта.

8. Передача параметрів за посиланням, а не за значенням (при передачі параметрів за посиланням можливе збереження параметрів у зовнішній пам’яті після перевірки їх коректності, порушник може змінити ці дані до їх використання).

9. Система може містити чимало елементів (наприклад, програм), що мають різні привілеї. Основною проблемою забезпечення безпеки операційної системи є проблема створення механізмів контролю доступу до ресурсів системи. Процедура контролю доступу полягає у перевірці відповідності запиту суб'єкта наданим йому правам доступу до ресурсів. Крім того, операційна система містить допоміжні засоби захисту, такі як засоби моніторингу, профілактичного контролю та аудиту. У сукупності механізми контролю доступу та допоміжні засоби захисту утворюють механізми управління доступом.

Засоби профілактичного контролю необхідні для відсторонення користувача від безпосереднього виконання критичних із точки зору безпеки даних операцій і передачі цих операцій під контроль операційної системи. Для забезпечення безпеки даних робота з ресурсами системи здійснюється за допомогою спеціальних програм операційної системи, доступ до яких обмежений.

Засоби моніторингу здійснюють постійне ведення реєстраційного журналу, в який заносяться записи про всі події в системі. В операційній системі можуть використовуватися засоби сигналізації про несанкціонований доступ, які використовуються при виявленні порушення безпеки даних або спроб порушення.

Контроль доступу до даних. При створенні механізмів контролю доступу необхідно, насамперед, визначити множину суб'єктів і об'єктів доступу. Суб'єктами можуть бути користувачі, задачі, процеси та процедури.

Об'єкти – це файли, програми, семафори, директорії, термінали, канали зв'язку, пристрої, блоки оперативної пам’яті тощо. Суб'єкти можуть одночасно розглядатися і як об'єкти, тому у суб'єкта можуть бути права на доступ до іншого суб'єкта. У конкретному процесі в даний момент часу суб'єкти є активними елементами, а об'єкти - пасивними.

Для здійснення доступу до об'єкта суб'єкт повинен володіти відповідними повноваженнями. Повноваження є певним символом, володіння яким дає суб'єкту певні права доступу до об'єкта. Область захистувизначає права доступу деякого суб'єкта до множини об'єктів, що захищаються, і є сукупністю всіх повноважень даного суб'єкта.

При функціонуванні системи необхідно мати можливість створювати нові суб'єкти та об'єкти. При створенні об'єкта одночасно створюється і повноваження суб'єктів із використання цього об'єкта. Суб'єкт, який створив таке повноваження, може скористатися ним для здійснення доступу до об'єкта або ж може створити кілька копій повноваження для передачі їх іншим суб'єктам.

З традиційної точки зору засоби управління доступом дозволяють специфікувати і контролювати дії, які суб'єкти (користувачі та процеси) можуть виконувати над об'єктами (інформацією та іншими комп'ютерними ресурсами).

Зараз піде мова про логічне управлінні доступом, яке, на відміну від фізичного, реалізується програмними засобами. Логічне управління доступом - це основний механізм багатокористувацьких систем, покликаний забезпечити конфіденційність і цілісність об'єктів і, деякою мірою, їх доступність (шляхом заборони обслуговування неавторизованих користувачів).

Розглянемо формальну постановку задачі в традиційному трактуванні. Є сукупність суб'єктів і набір об'єктів. Завдання логічного управління доступом полягає в тому, аби для кожної пари "суб'єкт-об'єкт" визначити множину допустимих операцій і контролювати виконання встановленого порядку.

Відношення "суб'єкти-об'єкти" можна подати у вигляді матриці доступу, в рядках якої перераховані суб'єкти, у стовпчиках - об'єкти, а в клітинах, розташованих на перетині рядків і стовпчиків, записані додаткові умови (наприклад, час і місце дії) і дозволені види доступу.

Тема логічного управління доступом - це одна з найскладніших в області інформаційної безпеки. Справа в тому, що саме поняття об'єкта (а тим більше видів доступу) змінюється від сервісу до сервісу. Для операційної системи до об'єктів відносяться файли, пристрої та процеси. Відносно файлів і пристроїв, як правило, розглядаються права на читання, запис, виконання (для програмних файлів), іноді на видалення і додавання. Окремим правом може бути можливість передачі повноважень доступу іншим суб'єктам (так зване право володіння). Процеси можна створювати і знищувати.

Сучасні операційні системи можуть підтримувати й інші об'єкти.

Для систем управління базами даних об'єкт - це база даних, таблиця, подання, збережена процедура. До таблиць застосовні операції пошуку, додавання, модифікації і видалення даних. У результаті при заданні матриці доступу потрібно брати до уваги не тільки принцип розподілу привілеїв для кожного сервісу, а й існуючі зв'язки між сервісами (доводиться дбати про узгодженість різних частин матриці). Аналогічна

трудність виникає при експорті/імпорті даних, коли інформація про права доступу, як правило, втрачається (оскільки на новому сервісі вона не має сенсу). Отже, обмін даними між різними сервісами має особливу небезпеку з точки зору управління доступом, а при проектуванні та реалізації різнорідної конфігурації необхідно подбати про узгоджений розподіл прав доступу суб'єктів до об'єктів і про мінімізацію числа способів експорту/імпорту даних.

Матрицю доступу, через її розрідженість (більшість клітин - порожні), нерозумно зберігати у вигляді двовимірного масиву. Як правило, її зберігають за стовпчиками, тобто для кожного об'єкта підтримується список "допущених" суб'єктів разом із їх правами. Елементами списків можуть бути імена груп і шаблони суб'єктів, що є великою підмогою адміністратору. Деякі проблеми виникають тільки при видаленні суб'єкта, коли доводиться видаляти його ім'я з усіх списків доступу; втім, ця операція проводиться нечасто.

Списки доступу – це виключно гнучкий засіб. За їх допомогою легко виконати вимогу про гранулярність прав із точністю до користувача. За допомогою списків нескладно додати права або явним чином заборонити доступ (наприклад, аби покарати кількох членів групи користувачів). Безумовно, списки є кращим засобом довільного управління доступом.

Переважна більшість операційних систем і систем управління базами даних реалізують довільне управління доступом. Основна перевага довільного управління – це гнучкість. На жаль, у "довільного" підходу є ряд недоліків. Розосередженість управління доступом веде до того, що довіреними повинні бути чимало користувачів, а не тільки системні оператори чи адміністратори. Через неуважність або некомпетентність співробітника, який володіє секретною інформацією, цю інформацію можуть дізнатися і всі інші користувачі. Отже, довільність управління повинна бути доповнена жорстким контролем за реалізацією вибраної політики безпеки.

Другий недолік, який є основним, полягає в тому, що права доступу існують окремо від даних. Ніщо не заважає користувачеві, який має доступ до секретної інформації, записати її в доступний усім файл або замінити корисну утиліту її "троянським" аналогом. Подібна "роздільність" прав і даних істотно ускладнює проведення декількома системами узгодженої політики безпеки і, головне, робить практично неможливим ефективний контроль узгодженості.

Повертаючись до питання подання матриці доступу, зазначимо, що для цього можна використовувати також функціональний спосіб, коли матрицю не зберігають в явному вигляді, а кожного разу обчислюють вміст відповідних клітин. Наприклад, при примусовому управлінні доступом застосовується порівняння міток безпеки суб'єкта та об'єкта.

Зручною надбудовою над засобами логічного управління доступом є обмежуючий інтерфейс, коли користувача позбавляють можливості спробувати здійснити несанкціоновані дії, включивши в число видимих йому об'єктів тільки ті, до яких він має доступ. Подібний підхід, як правило, реалізують у рамках системи меню (користувачеві показують лише допустимі варіанти вибору) або за допомогою обмежуючих оболонок, таких як restricted shell в ОС Unix.

При прийнятті рішення про надання доступу, як правило, аналізується така інформація:

1) ідентифікатор суб'єкта (код користувача, мережева адреса комп'ютера тощо). Подібні ідентифікатори є основою довільного (або дискреційного) управління доступом;

2) атрибути суб'єкта (мітка безпеки, група користувача і т.п.). Мітки безпеки – це основа мандатного управління доступом.

Безпосереднє управління правами доступу здійснюється на основі однієї з моделей доступу:

• матричної моделі доступу (модель Харрісона-Руззо-Ульмана);

• багаторівневої моделі доступу (модель Белла-Лападули).

Розробка та практична реалізація різних захищених ОС привела Харрісона, Руззо і Ульмана до побудови формальної моделі захищених систем. Схема моделі Харрісона-Руззо-Ульмана (HRU-моделі) наведена на рис. 5.1.

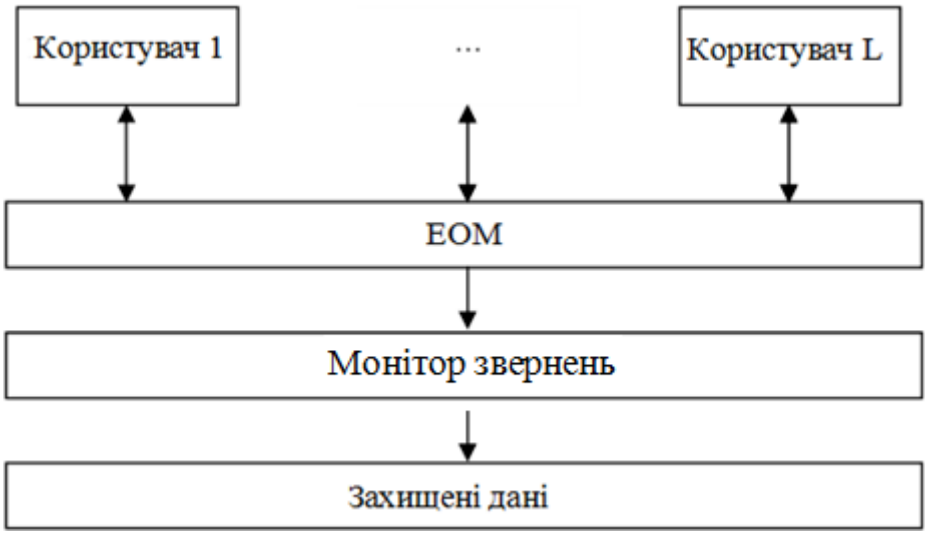

Рис. 5.1. Схема моделі Харрісона, Руззо і Ульмана

Модель Белла-Лападули описується скінченим автоматом із допустимим набором станів, в яких може перебувати інформаційна система. Кожному суб'єкту присвоюється свій рівень доступу, відповідний мірі конфіденційності. Аналогічно, об'єкту присвоюється рівень секретності. Поняття захищеної системи визначається таким чином: кожний стан системи повинен відповідати політиці безпеки, встановленої для даної інформаційної системи. Перехід між станами описується функціями переходу. Система знаходиться в безпечному стані в тому випадку, якщо у кожного суб'єкта є доступ тільки до тих об'єктів, до яких дозволений доступ на основі поточної політики безпеки. Для визначення, чи має суб'єкт права на одержання певного виду доступу до об'єкта, рівень секретності суб'єкта порівнюється з рівнем секретності об'єкта, і на основі цього порівняння вирішується питання, надати чи ні запитуваний доступ. Набори *рівень доступу/рівень секретності* описуються за допомогою матриці доступу.

Аналіз захищеності сучасних операційних систем

Зупинимося на принциповому, навіть, можна сказати, концептуальному протиріччі між реалізованими в ОС механізмами захисту та необхідними формалізованими вимогами. Концептуальному в тому розумінні, що це протиріччя характеризує не якийсь один механізм захисту, а загальний підхід до побудови системи захисту.

Суперечність полягає в принциповій відмінності підходів до побудови схеми адміністрування механізмів захисту і, як наслідок, це докорінно позначається на формуванні загальних принципів задання та реалізації політики безпеки в організації, розподілу відповідальності за захист інформації, а також на визначенні того, кого відносити до потенційних зловмисників (від кого захищати інформацію).

Для ілюстрації із сукупності формалізованих вимог до системи захисту конфіденційної інформації розглянемо такі дві вимоги:

1) право змінювати правила розмежування доступу (ПРД) має надаватися виділеним суб'єктам (адміністрації, службі безпеки тощо);

2) повинні бути передбачені засоби управління, що обмежують поширення прав на доступ.

Дані вимоги жорстко регламентують схему (або модель) адміністрування механізмів захисту. Це повинна бути централізована схема, єдиним елементом якої виступає виділений суб'єкт, зокрема, адміністратор (адміністратор безпеки). При цьому кінцевий користувач виключений у принципі зі схеми адміністрування механізмів захисту.

При реалізації концепції побудови системи захисту, регламентованої такими вимогами, користувач не наділяється елементом довіри, оскільки він може вважатися потенційним зловмисником, що і має місце на практиці.

Тепер у загальних рисах розглянемо концепцію, реалізовану в сучасних універсальних ОС. Тут "власником" файлового об'єкта, тобто особою, яка одержує право на задання атрибутів (або ПРД) доступу до файлового об'єкту, є особа, що створює файловий об'єкт. Файлові об'єкти створюють кінцеві користувачі, тому саме вони і призначають ПРД до створюваних ними файлових об'єктів. Іншими словами, в ОС реалізується розподілена схема призначення ПРД, де елементами схеми адміністрування є власне кінцеві користувачі.

У даній схемі користувач повинен наділятися практично такою ж довірою, як і адміністратор безпеки, при цьому нести поряд із ним відповідальність за забезпечення комп'ютерної безпеки. Відзначимо, що дана концепція реалізується і більшістю сучасних застосувань, зокрема СУБД, де користувач може поширювати свої права на доступ до ресурсів, що захищаються. Крім того, не маючи в повному обсязі механізмів захисту комп'ютерної інформації від кінцевого користувача, в рамках даної концепції неможливо розглядати користувача як потенційного зловмисника. А як побачимо далі, саме з несанкціонованими діями користувача

(як свідомими, так і ні) на комп'ютері, що захищається, пов'язана велика частина загроз комп'ютерної безпеки.

Відзначимо, що централізована і розподілена схеми адміністрування - це дві діаметрально протилежні точки зору на захист, що вимагають абсолютно різних підходів до побудови моделей та механізмів захисту.При цьому скільки-небудь гарантований захист інформації можна реалізувати тільки при прийнятті концепції повністю централізованої схеми адміністрування, що підтверджується відомими погрозами ОС.

Можливості моделей, методів і засобів захисту будемо розглядати стосовно до реалізації саме концепції централізованого адміністрування. Одним із елементів цієї концепції є розгляд користувача як потенційного зловмисника, здатного здійснити НСД до інформації, що захищається.

3. Основні вбудовані механізми захисту ОС та їх недоліки

Коротко зупинимося на основних механізмах захисту, вбудованих у сучасні універсальні операційні системи (ОС). Зробимо це стосовно можливості реалізації ними прийнятої нами для розгляду концепції захисту конфіденційної інформації.

Основні захисні механізми ОС ряду UNIX

Захист ОС ряду Unix у загальному випадку базується на трьох основних механізмах:

1) ідентифікації та аутентифікація користувача при вході в систему;

2) розмежуванні прав доступу до файлової системи, в основі якої знаходиться реалізація дискреційної моделі доступу;

3) аудит, тобто реєстрація подій.

При цьому відзначимо, що для різних клонів ОС ряду Unix можливості механізмів захисту можуть незначно відрізнятися, проте будемо розглядати ОС Unix у загальному випадку, без урахування деяких незначних особливостей окремих ОС цього ряду.

Побудова файлової системи і розмежування доступу до файлових об'єктів має особливості, притаманні даному ряду ОС. Розглянемо коротко ці особливості. Всі дискові накопичувачі (томи) об'єднуються в єдину віртуальну файлову систему шляхом операції монтування тому. При цьому вміст тому проектується на вибраний каталог файлової системи. Елементами файлової системи є також всі пристрої, що вмикаються до комп'ютера, який захищається (монтовані до файлової системи). Тому розмежування доступу до них здійснюється через файлову систему.

Кожний файловий об'єкт має індексний дескриптор, в якому серед іншого зберігається інформація про розмежування доступу до даного файлового об'єкту. Права доступу діляться на три категорії: доступ для власника, доступ для групи і доступ для інших користувачів. У кожній категорії визначаються права на читання, запис і виконання (у випадку каталогу - перегляд).

Користувач має унікальний символьний ідентифікатор (ім'я) і числовий ідентифікатор (UID). Символьний ідентифікатор пред'являється користувачем при вході в систему, числовий використовується операційною системою для визначення прав користувача у системі (доступ до файлів тощо).

Принципові *недоліки* захисних механізмів ОС ряду Unix. Розглянемо в загальному випадку недоліки реалізації системи захисту ОС ряду Unix у частині невиконання вимог до захисту конфіденційної інформації, безпосередньо пов'язаних із можливістю несанкціонованого доступу до інформації.

Для початку зазначимо, що в ОС ряду Unix внаслідок реалізованої нею концепції адміністрування (не централізованої) неможливо забезпечити замкнутість (або цілісність) програмного середовища. Це пов'язано з неможливістю встановлення атрибуту "виконання" на каталог (для каталогу даний атрибут обмежує можливість "огляду" вмісту каталогу). Тому при розмежуванні адміністратором доступу користувачів до каталогів користувач, як "власник" створюваного ним файлу, може занести у свій каталог виконуваний файл і, як його "власник", встановити на файл атрибут "виконання", після чого завантажити записану ним програму. Ця проблема безпосередньо пов'язана з реалізованою в ОС концепцією захисту інформації.

Не в повному обсязі реалізується дискреційна модель доступу, зокрема, не можуть розмежовуватися права доступу для користувача "root" (UID=0), тобто даний суб'єкт доступу виключається зі схеми управління доступом до ресурсів. Відповідно всі процеси, які запускаються, мають необмежений доступ до ресурсів, що захищаються. З цим недоліком системи захисту пов'язана множина атак, зокрема:

- несанкціоноване одержання прав root;
- запуск із правами root власного виконуваного файлу (локально чи віддалено впровадженого), при цьому несанкціонована програма одержує повний доступ до ресурсів, що захищаються, тощо.

Крім того, в ОС ряду Unix неможливо вбудованими засобами гарантовано видаляти залишкову інформацію. Для цього в системі абсолютно відсутні відповідні механізми.

Необхідно також відзначити, що більшість ОС даного ряду не мають можливості контролю цілісності файлової системи, тобто не містять відповідних вбудованих засобів. У кращому випадку додатковими утилітами може бути реалізований контроль конфігураційних файлів ОС за розкладом у той час, як найважливішою можливістю даного механізму можна вважати контроль цілісності програм (застосувань) передїх запуском, контроль файлів даних користувача і т.д.

Що стосується реєстрації (аудиту), то в ОС ряду Unix не забезпечується реєстрація видачі документів на "тверду копію", а також деякі інші вимоги до реєстрації подій.

Якщо ж трактувати вимоги до управління доступом у загальному випадку, то при захисті комп'ютера в складі ЛОМ необхідне управління доступом до вузлів мережі. Проте вбудованими засобами в деяких ОС ряду

Unix управління доступом до вузлів не реалізується.

З наведеного аналізу видно, що чимало механізмів, необхідних із точки зору виконання формалізованих вимог, більшістю ОС ряду Unix не реалізується в принципі, або реалізується лише частково.

Основні захисні механізми ОС ряду WINDOWS

Тепер коротко зупинимося на основних механізмах захисту, реалізованих в ОС ряду Windows, і проведемо аналіз захищеності цих операційних систем. Відзначимо, що тут ряд об'єктів доступу (зокрема, пристрої, реєстр ОС тощо) не є об'єктами файлової системи. Тому виникає питання, як варто трактувати таку вимогу "Система захисту повинна контролювати доступ іменованих суб'єктів (користувачів) до іменованих об'єктів (файлів, програмам, томів тощо)". Не зрозуміло, чи є об'єктами доступу, до яких, слідуючи формальним вимогам, необхідно розмежовувати доступ користувачів, наприклад, реєстр ОС і т.д.

На відміну від ряду ОС Unix, де всі завдання розмежувальної політики доступу до ресурсів вирішуються засобами управління доступом до об'єктів файлової системи, доступ у даних ОС розмежовується власним механізмом для кожного ресурсу. Іншими словами, при розгляді механізмів захисту ОС Windows постає завдання визначення та задання вимог до повноти розмежувань (це визначається тим, що вважати об'єктом доступу).

Так, як і для ряду ОС Unix, тут основними механізмами захисту є:

1) ідентифікація та аутентифікація користувача при вході в систему;

2) розмежування прав доступу до ресурсів, в основі якого знаходиться реалізація дискреційної моделі доступу (окремо до об'єктів файлової системи, до пристроїв, до реєстру ОС, до принтерів та ін.);

3) аудит, тобто реєстрація подій.

Тут явно виділяються (в кращу сторону) можливості розмежувань прав доступу до файлових об'єктів (для NTFS) - істотно розширені атрибути доступу, що встановлюються на різні ієрархічні об'єкти файлової системи (логічні диски, каталоги, файли). Зокрема, атрибут "виконання" може встановлюватися і на каталог, тоді він успадковується відповідними файлами.

При цьому істотно обмежені можливості управління доступом до інших ресурсів, що захищаються, зокрема, до пристроїв введення. Наприклад, тут відсутній атрибут "виконання", тобто неможливо заборонити запуск несанкціонованої програми з пристроїв введення.

Принципові *недоліки* захисних механізмів ОС ряду Windows. Насамперед розглянемо принципові недоліки захисту ОС ряду Windows, безпосередньо пов'язані з можливістю несанкціонованого доступу до інформації.

При цьому на відміну від ОС ряду Unix в ОС Windows неможлива в загальному випадку реалізація централізованої схеми адміністрування механізмів захисту або відповідних формалізованих вимог. Згадаймо, що в ОС Unix це поширювалося лише на завантаження процесів. Пов'язано це з тим, що в ОС Windows прийнята інша концепція реалізації розмежувальної політики доступу до ресурсів (для NTFS).

В рамках цієї концепції розмежування для файлу пріоритетніше, ніж для каталогу, а в загальному випадку - розмежування для файлового об'єкта, який включається, пріоритетніше, ніж для об'єкта, який включає. Це призводить до того, що користувач, створюючи файл і будучи його "власником", може призначити будь-які атрибути доступу до такого файлу (тобто дозволити до нього доступ будь-якому іншому користувачеві). Звернутися до цього файлу може користувач (якому призначив права доступу "власник") незалежно від встановлених адміністратором атрибутів доступу на каталог, в якому користувач створює файл. Ця проблема безпосередньо пов'язана з реалізованою в ОС Windows концепцією захисту інформації.

Далі, в ОС ряду Windows не в повному обсязі реалізується дискреційна модель доступу, зокрема, не можуть розмежовуватися права доступу для користувача "Система". В ОС присутні не тільки користувацькі, а йсистемні процеси, які завантажуються безпосередньо системою. При цьому доступ системних процесів не може бути розмежований. Відповідно, всі завантажені системні процеси мають необмежений доступ до ресурсів, які захищається. З цим недоліком системи захисту пов'язана множина атак, зокрема, несанкціоноване завантаження власного процесу з правами системного. До речі, це можливо і внаслідок некоректної реалізації механізму забезпечення замкнутості програмного середовища.

В ОС ряду Windows неможливо в загальному випадку забезпечити замкнутість (або цілісність) програмного середовища. Це пов'язано з іншими проблемами, ніж в ОС ряду Unix, в яких неможливо встановити атрибут "виконання" на каталог. Для з'ясування складності даного питання розглянемо два способи, якими в загальному випадку можна реалізувати даний механізм, причому обидва способи неспроможні. Отже, механізм замкнутості програмного середовища в загальному випадку може бути забезпечений:

- заданням списку дозволених до запуску процесів із наданням можливості користувачам запускати процеси тільки з цього списку. При цьому процеси задаються повношляховими іменами, причому засобами розмежування доступу забезпечується неможливість їх модернізації користувачем. Такий підхід не реалізується вбудованими в ОС механізмами;

- дозволом запуску користувачами програм тільки із заданих каталогів при неможливості модернізації цих каталогів. Однією з умов коректної реалізації даного підходу є заборона користувачам запуску програм інакше, ніж із відповідних каталогів. Некоректність реалізації ОС Windows даного підходу пов'язана з неможливістю встановлення атрибуту "виконання" на пристрої введення (дисковод або CD-ROM). У зв'язку з цим при розмежуванні доступу користувач може завантажити несанкціоновану програму з дискети, або з диска

CD-ROM (досить поширена атака на ОС даного ряду).

Тут же варто відзначити, що з точки зору забезпечення замкнутості програмного середовища [тобто реалізації механізму, який забезпечує можливість користувачам запускати тільки санкціоновані процеси (програми)] дії користувача із запуску процесу можуть бути як явними, так і прихованими.

Явні дії передбачають запуск процесів (виконуваних файлів), які однозначно ідентифікуються своїм іменем. Приховані дії дозволяють здійснювати вбудовані в застосування інтерпретатори команд. Прикладом таких є офісні застосування. При цьому прихованою діє користувача буде запуск макросу.

У даному випадку ідентифікації підлягає лише власне застосування, наприклад, процес winword.exe. При цьому процес може крім своїх регламентованих дій виконувати ті приховані дії, які задаються макросом (відповідно, ті, які допускаються інтерпретатором), що зберігається у відкритому документі. Те ж стосується і будь-якої віртуальної машини, що містить вбудований інтерпретатор команд. При цьому відзначимо, що при використанні застосувань, які мають вбудовані інтерпретатори команд (у тому числі офісних застосувань), не в повному обсязі забезпечується виконання вимоги щодо ідентифікації програм.

Повертаючись до обговорення недоліків, відзначимо, що в ОС ряду Windows неможливо вбудованими засобами гарантовано видаляти залишкову інформацію. В системі відсутні відповідні механізми.

Крім того, ОС ряду Windows не має в повному обсязі можливість контролю цілісності файлової системи. Вбудовані механізми системи дозволяють контролювати лише власні системні файли, не забезпечуючи контроль цілісності файлів користувача. Крім того, вони не вирішують найважливіше завдання даних механізмів - контроль цілісності програм (застосувань) перед їх запуском, контроль файлів даних користувача та ін.

Що стосується реєстрації (аудиту), то в ОС ряду Windows не забезпечується реєстрація видачі документів на "тверду копію", а також деякі інші вимоги до реєстрації подій. Знову ж таки, якщо трактувати вимоги до управління доступом у загальному випадку, то при захисті комп'ютера в складі ЛОМ необхідне управління доступом до вузлів мережі (розподілений пакетний фільтр). В ОС ряду Windows механізм управління доступу до вузлів у повному обсязі не реалізується.

Що стосується мережевих ресурсів, то фільтрації піддається тільки доступ до вхідного ресурсу, а запит доступу на комп'ютері, з якого він здійснюється, фільтрації не підлягає. Це принципово, оскільки не можуть підлягати фільтрації застосування, якими користувач здійснює доступ до ресурсів. Завдяки цьому, досить поширеними є атаки на протокол NETBIOS.

Крім того, в повному обсязі управляти доступом до ресурсів можна тільки при встановленій на всіх комп'ютерах ЛОМ файловій системі NTFS. В іншому випадку неможливо заборонити запуск несанкціонованої програми з віддаленого комп'ютера, тобто забезпечити замкнутість програмного середовища в цій частині.

З наведеного аналізу можна зробити висновок, що чимало механізмів, необхідних із точки зору виконання формалізованих вимог із захисту, ОС ряду Windows не реалізують у принципі, або реалізують лише частково.

З урахуванням сказаного можемо зробити важливий висновок щодо того, що більшістю сучасних універсальних ОС не виконуються в повному обсязі вимоги до захисту автоматизованих систем. Це означає, що вони не можуть без використання додаткових засобів захисту застосовуватися для захисту навіть конфіденційної інформації. При цьому слід зазначити, що основні проблеми захисту тут викликані не неможливістю ОС до вимог окремих механізмів захисту, а принциповими причинами, зумовленими реалізованою в ОС концепцією захисту. Концепція ця ґрунтується на реалізації розподіленої схеми адміністрування механізмів захисту, що само по собі є невиконанням формалізованих вимог до основних механізмів захисту.

Теми доповідей

1. Призначення та класифікація програмного забезпечення ПК.
2. Поняття файлу. Правила іменування файлів. Групові імена файлів. Каталоги (папки). Поняття шляху до файлу.
3. Поняття про операційну систему комп’ютера. Розповсюджені ОС.
4. Класифікація сучасних операційних систем. Багатозадачні операційні системи та віконний інтерфейс.
5. Основні поняття та об‘єкти операційної системи сімейства Windows.
6. Призначення файлових менеджерів.
7. Поняття про комп'ютерні віруси. Принципи “зараження” комп'ютерним вірусом.
8. Профілактика та лікування вірусів. Антивірусні програми.
9. Архівація файлів. Основні можливості програм-архваторів

Література:

1. Зацеркляний М.М. Компютерні основи систем кібербезпеки: навч. посібник/ Зацеркляний М.М., Струков В.М.-Харків: Тов. «В деле», 2017.- 292 с.
2. Інформатика: Комп’ютерна техніка. Комп’ютерні технології: Підручник для студентів вищих навчальних закладів / За ред. Пушкаря О.І. – К.: Каравела, 2004.
3. Інформатика та інформаційні технології. — Харків. ООО Компанія Сміт , 2007. — 352 с.

4. Основи інформаційних технологій і систем- Підручник / В. А. Павлиш, Л. К. Гліненко, Н. Б. Шаховська. Львів : Видавництво Львівської політехніки, 2018. 620 с

Рекомендована література (основна, додаткова), інформаційні та навчальні ресурси в Інтернеті

Основна література

1. Основи математичної логіка: навчальний посібник/Зубенко В.В., Шкільняк С.С.. К.: НУБіП України, 2020. -102 с.
2. Дискретна математика : навч. посіб. / уклад. : С. І. Балога; рец. : О. О. Погоріляк. – Ужгород : ПП «АУТДОР-ШАРК», 2021. – 124 с.
3. Нікольський, Ю. В. Дискретна математика : підручник / Ю. В. Нікольський, В. В. Пасічник, Ю. М. Щербина. – Львів : Магнолія, 2018. – 432 с.
4. Математична логіка та теорія алгоритмів: Лекції: навч. посіб. для студ. спеціальності 124 «Системний аналіз» / О. В. Стусь ; КПІ ім. Ігоря Сікорського. 2017. – 150 с.
5. Логічна алгебра: методичний посібник./ Н.А. Якімова. – Одеса: «Освіта України», 2019. – 40 с.
6. Математична логіка та теорія алгоритмів: Навчальний посібник/З.П. Халецька, В.В. Нарадовий. – Кропивницький: РВВ КДПУ ім.. В. Винниченка, 2017. – 128 с.
7. Архітектура комп'ютерів. Частина 1 : лабораторний практикум / Л. В. Крупельницький, А. В. Снігур, С. В. Богомолов. – Вінниця : ВНТУ, 2020. – 104 с.
8. Архітектура комп'ютера та конфігурування комп'ютерних систем (на основі фундаменталізованого підходу) : навч. посіб. /Антоненко О. В., Бардус І. О.− Бердянськ :2018 – 292 с.
9. Архітектура комп'ютерів та периферійні пристрої: Навч. посібник / С. Є. Бантюков, О. В. Чаленко, В. С. Меркулов та ін. – Харків: УкрДУЗТ, 2018. – Ч. 1. – 116 с.
10. Архітектура комп'ютера. Навчальний посібник./ Матвієнко М. П., Розен В. П., Закладний О. М. — К: Видавництво Ліра-К, 2016. — 264 с.
11. Зацеркляний М.М. Компютерні основи систем кібербезпеки: навч. посібник/ Зацеркляний М.М., Струков В.М.-Харків: Тов. «В деле», 2017.- 292 с.

Додаткова література

1. Зацеркляний М.М. Інформаційні системи і технології в діяльності правоохоронних
2. Основи інформаційних технологій і систем- Підручник / В. А. Павлиш, Л. К. Гліненко, Н. Б. Шаховська. Львів : Видавництво Львівської політехніки, 2018. 620 с
3. Матвієнко М.П., Шаповалов С.П. Математична логіка та теорія алгоритмів. Навчальний посібник. – КК.: Видавництво Ліра-К, 2017. – 212 с.
4. Матвієнко М.П., Шаповалов С.П. Математична логіка та теорія алгоритмів. Навчальний посібник. — Математичний практикум. — Київ : Ліра-К, 2015. — 212 с
5. Тарарака В.Д. Архітектура комп'ютерних систем: навчальний посібник. – Житомир :ЖДТУ, 2018. – 383 с.
6. Schwichtenberg, Helmut (2003–2004). Mathematical Logic. Munich, Germany:Mathematisches Institut der Universität München. Процитовано 2016-06-14. (англ.)

Інформаційні ресурси в Інтернеті

1. Математична логіка. Практикум [Електронний ресурс]: навч. посіб. для студ. спеціальності 113 «Прикладна математика», освітньої програми «Наука про дані та математичне моделювання» / О.Л.Темнікова ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1,37 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 76 с. https://ela.kpi.ua/bitstream/123456789/42844/1/WorkshopLogicTemnikova.pdf

Перелік програмного забезпечення

1. Операційна система MS Windows 7-10 - для засвоєння правил роботи з системоювведення-виведення інформації та її зберігання на зовнішніх носіях.
2. Операційна оболонка (TotalCommander або аналогічна) - для засвоєння правил роботиз файлами, що зберігаються на носіях інформації.
3. Набір прикладних сервісних програм (антивірусні програми, архіватори).