3. СИСТЕМА КОМАНД ПРОЦЕСОРА IA-32

3.1. ФОРМАТ МАШИННИХ КОМАНД IA-32

При вивченні системи машинних команд необхідно враховувати дві складові – безпосередньо набір машинних команд та правила представлення цих команд на рівні процесора, тобто формат машинних команд. Машинні команди являють собою сформовані за визначеними правилами послідовності нулів та одиниць. Для того щоб примусити процесор виконати деяку дію, йому необхідно видати відповідну вказівку у вигляді машинної команди, а для виконання більш складної роботи достатньо написати програму у двійкових кодах. Програмування перших комп'ютерів здійснювалось саме так. Для полегшення процесу розробки програм було розроблено *Assembler* як символічний аналог машинної мови, а до архітектури комп'ютера було введено блок мікропрограмного управління. Для кожної машинної команди блок мікропрограмного управління містить окрему мікропрограму, за допомогою якої дії, що задані цією командою, переводяться на мову сигналів, які надходять до необхідних підсистем процесора. Після цих нововведень процес розробки програм значно спростився.

Існує взаємна однозначна відповідність машинних команд та *Assembler*. Розуміння правил формування машинних команд із команд *Assembler* є однією з необхідних умов розуміння логіки роботи комп'ютера.

Машинна команда являє собою закодовану за визначеними правилами вказівку мікропроцесору на виконання деякої операції. Правила кодування команд називають *форматом* команд. Команди процесорів архітектури IA-32 вважаються складними. Максимальна довжина машинної команди – 15 *байт*. Реально машинна команда може містити меншу кількість полів, аж до одного – операції. Загальний формат машинної команди наведено на рис. 3.1. Як на рівні формату співвідносяться між собою машинні команди та команди *Assembler*? Розглянуто приклад застосування команди переміщення:

```
mov ax, 9
mov bx, c
mov cx, bx
mov dl, cl
mov dh, ah
```

Результати трансляції наведено нижче:

```
16    0007    B8 0009          mov ax, 9
17    000A    8B 1E 0002r      mov bx, c
18    000E    8B CB            mov cx, bx
19    0010    8A D1            mov dl, cl
20    0012    8A F4            mov dh, ah
```

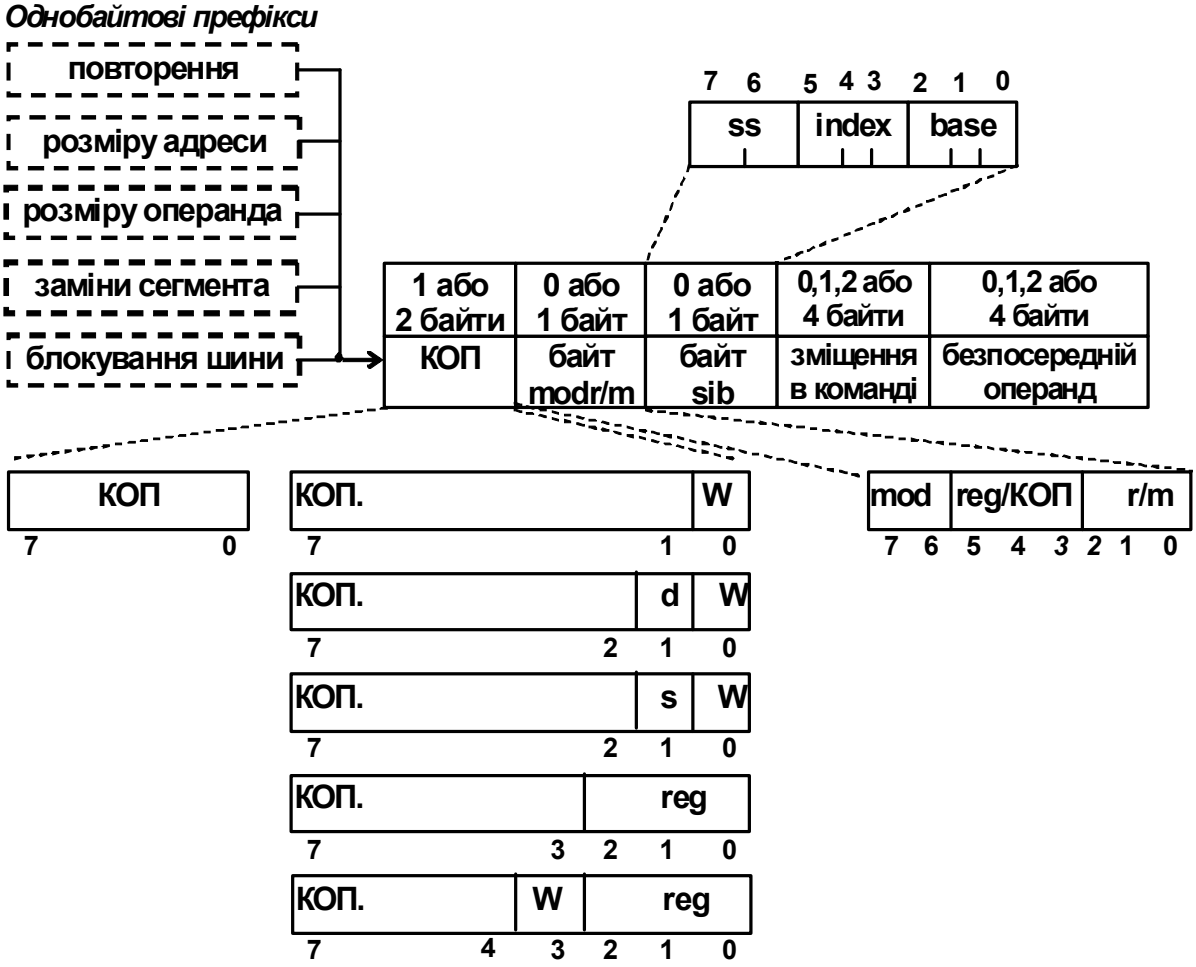

Рис. 3.1. Формат машинної команди

Значення B8, 8B, 8A – це код операції. Видно, що команда переміщення та сама, а коди машинних команд різні – B8, 8B, 8A. Більшість команд *Assembler* мають декілька можливих варіантів сполучення операндів. Як показано в прикладі, незважаючи на однакові назви команд *Assembler*, для кожного можливого сполучення операндів є своя машинна команда зі своїм значенням поля коду операції.

Логічно, що будь-яка команда *Assembler* містить декілька елементів. Елемент, який описує що робити, називається кодом операції (КОП). Значення в полі КОП деяким чином у блоці мікропрограмного управління визначає підпрограму, яка реалізує дії для даної команди.

Елемент, який описує об'єкти, з якими потрібно щось робити, є операндами. Операнди в команді можуть не задаватися, а використовуватись за замовчуванням.

Елементи, які описують, як робити, є типами операндів і, як правило, задаються неявно. Вони використовуються транслятором *Assembler* при формуванні машинної команди для визначення поля коду операції.

Ці самі елементи має й машинна команда, але в закодованому вигляді. Перетворення команд *Assembler* у відповідні машинні команди здійснює спеціальна програма – *Assembler*, яку також називають транслятором (компілятором) *Assembler*.

Розглянемо призначення полів машинної команди.

Префікси. *Префікси* – необов'язкові елементи машинної команди. Призначення префіксів – змінювати дії, які виконуються командою. Префікси можуть вказуватися програмістом явно при написанні початкового тексту програми, або їх може вставити *Assembler*. Процесор розпізнає префікси за їх значеннями. Машинна команда може мати до чотирьох префіксів одночасно. У пам'яті префікси передують команді. При цьому їх порядок слідування може бути будь-яким.

Типи префіксів, які може використовувати прикладна програма.

Префікс заміни сегмента в наявній формі вказує, який сегментний регістр використовується в даній команді для адресації стеку або даних. Префікс відміняє вибір сегментного регістра за замовчуванням. Префікси заміни сегмента мають такі значення:

2Eh – заміна сегмента CS;
36h – заміна сегмента SS;
3Eh – заміна сегмента DS;
26h – заміна сегмента ES;
64h – заміна сегмента FS;
65h – заміна сегмента GS.

Префікс повторювання використовується з командами обробки рядків (ланцюгові команди). Цей префікс "зациклює" команду для обробки всіх елементів ланцюга. Система команд підтримує два типи префіксів: *безумовні* (REP – 0F3h), які змушують команду повторюватись певну кількість разів, та *умовні* (REPE/REPZ – 0F3h, REPNE/REPNZ – 0F2h), які при зациклюванні перевіряють деякі прапорці та внаслідок перевірки можливий достроковий вихід із циклу.

Префікс блокування шини ініціює видачу процесором сигналу LOCK# (значення 0F0h) для блокування системної шини; використовується в багатопроцесорних системах для забезпечення монопольного володіння системою.

Префікс розрядності (розміру) адреси (значення 67h) уточнює розрядність адреси (32- або 16-розрядна). Кожній команді, у якій використовується адресний операнд, ставиться у відповідність розрядність адреси цього операнда. Якщо розрядність адреси для даної команди дорівнює 16 *біт*, то це означає, що команда містить 16-розрядне зміщення й воно відповідає 16-розрядному зміщенню адресного операнда відносно початку деякого сегмента. За допомогою цього префікса можна змінити значення розрядності, яке діє за замовчуванням. Ця зміна буде стосуватися лише тієї команди, якій передує цей префікс.

Префікс розрядності (розміру) операнда (значення 66h)– аналогічний до префікса розрядності адреси, однак указує на розрядність операндів (32- або 16-розрядні), з якими працює команда.

Значення атрибутів розрядності адреси та операндів встановлюється за замовчуванням. Якщо команда має операнд у пам'яті, то його адреса являє собою значення зміщення відносно початку сегмента даних (якщо не застосовується префікс переозначення сегмента) та міститься у полі зміщення машинної команди. Розмір цього поля залежить від поточного режиму адресації – атрибути *use16* або *use32* в директивах сегментації. При 16-розрядній адресації розмір поля зміщення в машинній команді дорівнює 16 *біт*. При 32-розрядній адресації розмір поля зміщення в машинній команді відповідає 32 *біт*. Явне означення префікса розміру адреси дозволяє вказати процесору значення, яке відрізняється від діючого за замовчуванням. Наприклад, якщо діючий розмір адреси дорівнює 16 *біт*, то застосування перед будь-якою командою префікса 67h визначить для неї (і тільки для неї) розмір адреси 32 *біт*. І навпаки, якщо діючий розмір адреси дорівнює 32 *біт*, то представлення перед командою префікса 67h визначить для неї (і тільки для неї) розмір адреси 16 *біт*. Фізично це буде впливати на розмір поля зміщення в даній машинній команді.

При роботі процесора в реальному режимі й режимі віртуального *i8086* атрибути розрядності адреси та операндів за замовчуванням дорівнюють 16 бітам. У захищеному режимі значення цих атрибутів залежить від стану біта D в дескрипторі сегмента коду. Якщо $D = 0$, то значення цих атрибутів становить 16 бітів, а якщо $D = 1$ – то 32 біти. Змінити діючі за замовчуванням атрибути адреси та розміру операндів можна застосуванням атрибутів *use16* та *use32* в директивах сегментації.

У реальному режимі за допомогою префікса розрядності адреси можна задіяти 32-розрядну адресацію, але при цьому необхідно

пам’ятати про обмеженість розміру сегмента значенням 64 *Кбайт*. Аналогічно до префікса розрядності адреси можна використовувати префікс розрядності операнда в реальному режимі для роботи з 32-розрядними операндами, наприклад в арифметичних командах.

У команді префікси розміру адреси та операнда можуть задаватися одночасно. У табл. 3.1 показано, як комбінація префіксів впливає на атрибути розміру операнда та адреси машинної команди.

Таблиця 3.1

Значення атрибутів розмірів операнда та адреси

Біт D	66h	67h	Розмір операнда	Розмір адреси
0	–	–	16	16
0	–	+	16	32
0	+	–	32	16
0	+	+	32	32
1	–	-	32	32
1	–	+	32	16
1	+	–	16	32
1	+	+	16	16

Код операції. *Код операції* – обов’язковий елемент, який описує операцію, що виконується командою. Код операції може займати від одного до трьох байтів. Для деяких машинних команд частина бітів коду операції може розміщуватись в байті modr/m.

Багатьом командам відповідає декілька КОП, кожен із яких визначає особливості виконання операції. Поле коду операції не має однозначної структури (див. рис. 3.1). Залежно від конкретних команд, не обов’язкових з погляду мови *Assembler*, воно може мати у своєму складі від одного до трьох елементів, призначення яких наведено в табл. 3.2.

Таблиця 3.2

Призначення додаткових бітів поля коду операції

Поле	Кількість бітів	Призначення
d	1	Визначає напрямок передачі даних: 0 – передача даних із регістра reg в пам’ять (або регістр), яка адресується полем r/m; 1 – передача даних із пам’яті (або регістра), яка адресується полем r/m в регістр reg. За наявності байта sib адреса операнда в пам’яті формується з урахуванням вмісту цього байта.

Закінчення табл. 3.2

Поле	Кількість бітів	Призначення
s	1	Задає необхідність розширення 8-розрядного безпосереднього операнда до 16 або 32 бітів. Старші біти при цьому заповнюються значеннями старшого (знакового) біта початкового 8-розрядного операнда
w	1	Визначає розмір даних, якими оперує команда: байт, слово, подвійне слово: 0–8 *біт*, 1–16 *біт* для 16-розрядного розміру операндів або 32 *біт* для 32-розрядного розміру операндів.
reg	3	Визначає регістр, який використовується в команді. Значення поля залежить від поля w навіть у тому випадку, коли поле w відсутнє.

Наступні поля машинної команди визначають характеристики та місце розміщення операндів, які беруть участь у операції. Розгляд цих полів пов'язаний зі способом завдання операндів у машинній команді й тому буде викладений нижче.

Байт режим адресації mod r/m. Байт режиму адресації mod r/m іноді називають *постбайтом*, містить інформацію про операнди та режим адресації. Більшість команд процесора *Intel* – двоопернадні. Операнди можуть розміщуватись в пам'яті, в одному або двох регістрах.

Архітектура IA-32 не допускає, щоб обидва операнди розміщувались в пам'яті. Якщо операнд розміщується в пам'яті, то байт mod r/m визначає компоненти (зміщення, базовий та індексний регістри), які використовуються для розрахунку його ефективної адреси. Байт mod r/m складається з трьох полів.

1. Поле mod (2 біти) визначає спосіб адресації та кількість байтів у команді, що займає адреса операнда (поле "зміщення в команді"). Застосовується разом із полем r/m, яке вказує спосіб модифікації адреси операнда "зміщення в команді". Поле mod разом із полем r/m утворюють 32 можливих значення, які позначають один із восьми регістрів та 24 режими адресації. Наприклад, якщо mod=00, то поле "зміщення в команді" відсутнє й адреса операнда визначається вмістом базового і/або індексного регістрів. Якщо mod=01, це означає, що поле зміщення в команді присутнє, займає один байт і модифікується вмістом базового і/або індексного регістрів. Якщо mod=10, це означає, що поле зміщення в команді присутнє, займає 2 або 4 байти (залежно від діючого за замовчуванням або визначеного префіксом розміру адреси) і модифікується вмістом базового і/або індексного

регістрів. Якщо mod=11, це означає, що операндів у пам'яті немає, вони знаходяться в регістрах. Таке саме значення байта mod використовується у випадку, коли в команді використовується безпосередній операнд.

2. Поле reg (3 біти) визначає або регістр, який існує в команді на місці *другого* операнда, або можливе розширення коду операції (утворюючи в сукупності розмір поля КОП 11 *біт*).

3. Поле r/m використовується разом з полем mod і визначає або регістр, який розміщується в команді на місці *першого* операнда (mod = 11), або використовується для розрахунку ефективної адреси (сумісно з полем "зміщення в команді").

У табл. 3.3 та 3.4 немає вмісту поля reg для 16-розрядних регістрів 32-розрядних операціях, через те, що в архітектурі *Intel* окремо використовувати старшу половину 32-розрядних регістрів неможливо. В архітектурі *Intel* один з операндів обов'язково розміщений у регістрі і він може бути першим або другим. Розміщення першого та другого операндів у команді у форматі команди фіксоване. Однак, наприклад, команда MOV може виконувати переміщення як із регістра в пам'ять, так і з пам'яті в регістр. У машинному представленні та сама команда. У її полі reg буде міститися код регістра, а в полі r/m – код режиму адресації. Ці дві команди будуть відрізнятися лише одним бітом d, який визначає напрямок передачі. Якщо в команді задіяні два регістри, то в цьому випадку діє правило: поле reg визначає *другий* операнд, а поле r/m – *перший*. Якщо команда MOV працює з коміркою пам'яті, то у вихідному тексті програми можуть бути такі варіанти представлення цієї команди:

```
mov  mem, ax
```

або

```
mov ax, mem
```

Таблиця 3.3

Значення кодів в полі reg (поле w присутнє в команді)

Поле reg	w=0	w=1
000	AL	AX/EAX
001	CL	CX/ECX
010	DL	DX/EDX
011	BL	BX/EBX
100	AH	SP/ESP
101	CH	BP/EBP
110	DH	SI/ESI
111	BH	DI/EDI

Таблиця 3.4

Значення кодів в полі reg (поле w відсутнє в команді)

Поле reg	16-розрядні операції	32-розрядні операції
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
110	SI	ESI
111	DI	EDI

У машинному представленні ці дві команди будуть виглядати однаково, із виключенням біта d. Для команди mov mem, ax біт d = 0, а для команди mov ax, mem біт d = 1.

Байт масштабу, індексу та бази (байт sib: *Scale – Index – Base*) використовується для розширення можливостей адресації операндів. На наявність байта sib у машинній команді вказує сполучення одного зі значень 01 або 10 поля mod і значення поля r/m = 100. Байт sib складається з трьох полів.

1. **Поле масштабу ss**. У цьому полі розміщується масштабний множник (значення 1, 2, 4 або 8) для індексного компонента index, який займає наступні три біти. При розрахунку ефективної адреси саме на це значення буде множитись вміст індексного регістра.

2. **Поле index** використовується для збереження номера індексного регістра, вміст якого також використовується для розрахунку ефективної адреси операнда.

3. **Поле base** використовується для збереження номера базового регістра, який також використовується для розрахунку ефективної адреси операнда. Як базовий та індексний регістри можуть використовуватись майже всі регістри загального призначення.

Поле зміщення в команді – це 8-, 16- або 32-розрядне число зі знаком, яке являє собою повністю або частково значення ефективної адреси операнда.

Поле безпосереднього операнда – необов'язкове поле, яке являє собою 8-, 16- або 32-розрядний безпосередній операнд. Наявність цього поля впливає на значення байта mod r/m.

3.2. ФУНКЦІОНАЛЬНА КЛАСИФІКАЦІЯ МАШИННИХ КОМАНД

Система машинних команд (рис. 3.2) найважливіша складова архітектури комп'ютера, яка визначає можливості його програмування. Система команд процесора *Pentium IV* архітектури IA-32 містить більше 300 команд. Весь набір команд можна розділити на чотири групи.

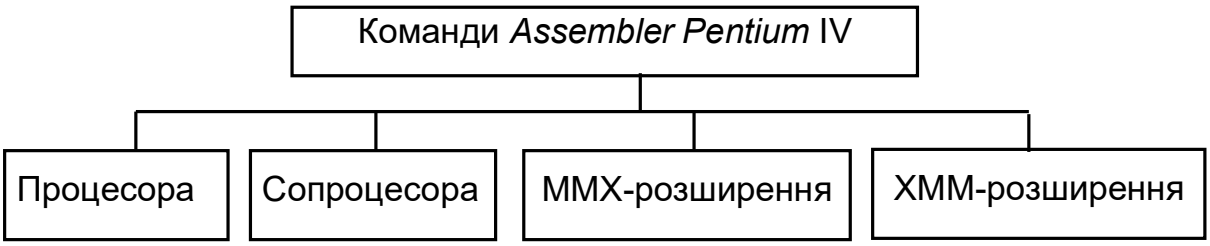

Рис. 3.2. Машинні команди процесора

Команди основного процесора (цілочислового) за функціональною ознакою класифікуються так, як показано на рис. 3.3.

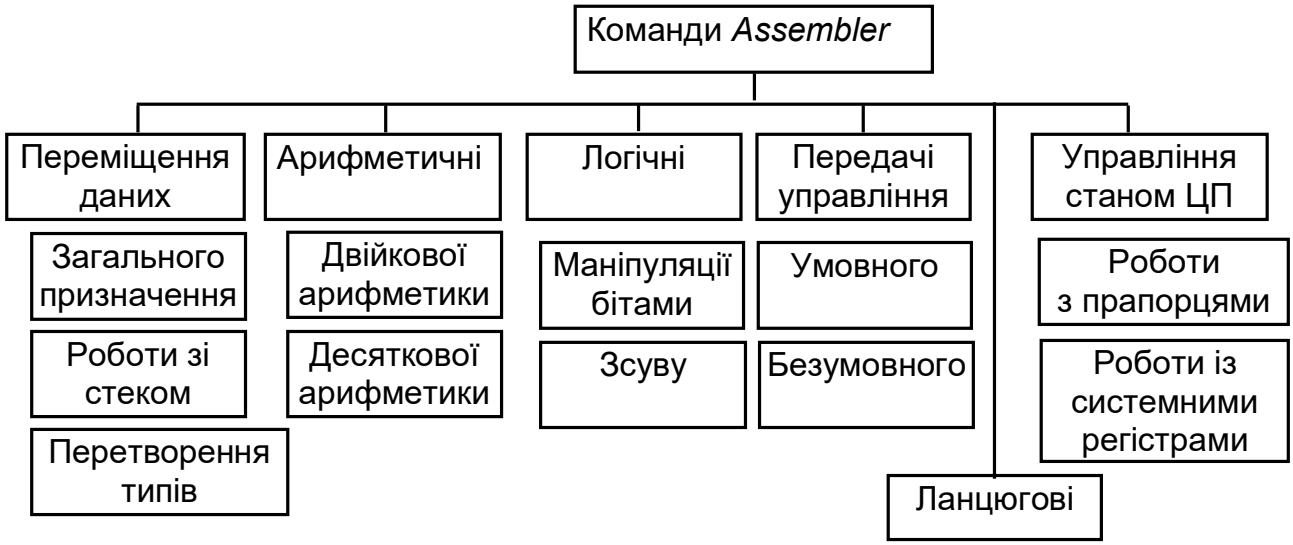

Рис. 3.3. Функціональна класифікація цілочислових машинних команд

Функціональну класифікацію інших груп команд буде наведено далі при вивченні відповідного матеріалу.

Контрольні питання

1. Назвати обов'язкові поля машинної команди.
2. Для чого призначений код операції?
3. Однобайтові префікси машинної команди. Для чого вони призначені?
4. Для чого призначений байт modr/m?
5. Для чого призначений байт sib?
6. Як класифікуються машинні команди процесора?