МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування
імені адмірала Макарова

Г. В. ПАВЛОВ, М. В. ПОКРОВСЬКИЙ

Методичні вказівки до виконання лабораторних робіт із дисципліни "МІКРОПРОЦЕСОРНА ТЕХНІКА"

Рекомендовано Методичною радою НУК

Миколаїв 2007

УДК 621.314

Павлов Г. В., Покровський М. В. Методичні вказівки до виконання лабораторних робіт із дисципліни "Мікропроцесорна техніка". – Миколаїв: НУК, 2007. – 36 с.

Кафедра комп'ютеризованих систем управління

Подано теоретичний матеріал, опис виконання, завдання та контрольні запитання для лабораторних робіт із програмування мовою асемблера *I*8086. У п'ятьох лабораторних роботах розглянуто використання практично всіх команд асемблера *I*8086: команд передачі даних, арифметичних, логічних операцій, зміщення даних, передачі управління, обробки блоків даних. Роботи виконуються за допомогою *IBM*-сумісного комп'ютера з установленою системою програмування, яка підтримує асемблер *I*8086.

Призначено для виконання лабораторних робіт студентами спеціальності 8.091401 "Системи управління і автоматики" з метою формування навичок програмування мікропроцесорних систем мовами низького рівня та вивчення принципів побудування структурованих програмних кодів.

Рецензент канд. техн. наук, доц. Д.В. Костенко

Лабораторна робота № 1

Тема: КОМАНДИ ПЕРЕДАЧІ ДАНИХ

Хід роботи

1. Розглянути набір команд передачі даних та вивчити особливості їх застосування.

2. Для закріплення отриманих теоретичних знань створити програму, використовуючи команди передачі даних. Для індикації результату використати вбудований динамік ПК. Варіанти створення програм для цієї і подальших робіт: використання налагоджувального середовища; використання тегів *Pascal asm ... end*; написання повноцінної програми мовою асемблера.

3. Підготуватися до захисту лабораторної роботи за наведеними запитаннями.

Теоретичні відомості

IN (*INput*) – ввід байта або слова. Прапори не змінюються. Команда: *IN accumulator, port*. Логіка: *accumulator* = (*port*). *IN* передає байт або слово із заданого порта (*port*) у регістр *AL* або *AX*. Адреса порта може визначатися як безпосередні байтовим значенням (у діапазоні 0...255), так і з використанням непрямої адресації за регістром *DX*.

Примітка. Потрібно вказати на те, що апаратна частина не використовує порти від F8*h* до FF*h* для вводу/виводу, бо вони зарезервовані для контролю за зовнішнім процесором та для інших можливих розширень процесора в майбутньому.

LAHF (*Load AH from Flags*) – завантаження *AH* із регістра прапорів. Прапори не змінюються. Команда: *LAHF*. Логіка: біти регістра *AH*: 7, 6, 4, 2, 0 відповідають бітам регістра прапорів *FLAGS*: *S*, *Z*, *A*, *P*, *C*. Команда *LAHF* копіює п'ять прапорів процесора 8080/8086 (прапори знака, нульового результату, допоміжного переносу, парності та переносу) у біти регістра *AH* з номерами 7, 6, 4, 2, 0 відповідно. Самі прапори при виконанні цієї команди не змінюються.

Примітка. Ця команда використовується, в основному, з метою забезпечення сумісності мікропроцесорів сімей 8080/8085 і 8086. Після виконання цієї команди значення бітів регістра *AH* з номерами 1, 3 і 5 не визначені.

LDS – завантаження покажчика із використанням *DS*. Прапори не змінюються. Команда: *LDS destination, source*. Логіка: *DS* = (*source*) *destination* = (*source* + 2). Команда *LDS* завантажує у два регістри 32-бітний покажчик, розташований у пам'яті за адресою *source*. При цьому старше слово заноситься у сегментний регістр *DS*, а молодше слово – у базовий регістр *destination*. Як операнд *destination* може виступати будь-який 16-бітний регістр, окрім сегментних.

Примітка. Команда *LES* – завантаження покажчика з використанням *ES* – виконує ті ж дії, що і *LDS*, але використовує при цьому замість регістра *DS* регістр *ES*.

LEA – завантаження виконавчої адреси. Прапори не змінюються. Команда: *LEA destination, source*. Логіка: *destination* = *Addr* (*source*). Команда *LEA* присвоює операнду *destination* значення зміщення (*offset*) операнда source (а не його значення!). Операнд source повинен бути посиланням на пам'ять, а як операнд *destination* може виступати будь-який 16-бітний регістр, окрім сегментних.

Примітка. Ця команда у порівнянні із використанням оператора *OFFSET* в команді *MOV* має ту перевагу, що операнд *source* може мати індекси. Наприклад, у наступній строці немає помилок: *LEA BX, TABLE* [*SI*] у той час, як строка *MOV BX, OFFSET TABLE* [*SI*] помилкова, оскільки оператор *OFFSET* обчислюється під час ассемблювання, а вказана адреса не буде відома до тих пір, поки програма не буде виконана.

LES – завантаження покажчика із використанням *ES*. Прапори не міняються. Команда: *LES destination, source*. Логіка: *ES* = (*source*) *destination* = (*source* + 2). Команда *LES* завантажує у два регістри 32-бітний покажчик, розташований у пам'яті за адресою *source*. При цьому старше слово заноситься в сегментний регістр *ES*, а молодше слово –

у базовий регістр *destination*. Як операнд *destination* може виступати будь-який 16-бітний регістр, крім сегментних.

Примітка. Команда *LDS* – завантаження покажчика із використанням *DS* – виконує ті ж дії, що й *LES*, але використовує при цьому замість регістра *ES* регістр *DS*.

MOV (*MOVe*) – пересилання (байта або слова). Прапори не міняються. Команда: *MOV destination*, *source*. Логіка: *destination* = *source*. *MOV* пересилає за адресою *destination* байт або слово, що перебувають за адресою *source*. Залежно від використовуваних режимів адресації команда *MOV* здійснює пересилання наступних видів:

із регістра загального призначення в регістр загального призначення;

із комірки пам'яті в регістр загального призначення;

із регістра загального призначення в комірку пам'яті;

безпосередній операнд у регістр загального призначення;

безпосередній операнд у комірку пам'яті;

із регістра загального призначення в сегментний регістр;

із сегментного регістра в регістр загального призначення;

із сегментного регістра в комірку пам'яті.

Примітка. Якщо операнди описані по-різному або режим адресації не дозволяє однозначно визначити розмір операнда, то для уточнення розміру переданих даних у команду варто включити один з операторів типу даних *BYTE PTR* або *WORD PTR*.

OUT (*OUTput*) – завантаження в порт. Прапори не міняються. Команда: *OUT port*, *accumulator*. Логіка: (*port*) = *accumulator*. *OUT* передає байт або слово з *AL* або *AX* у заданий порт. Адреса порта може визначатися як безпосереднім байтовим значенням (у діапазоні 0...255), так і з використанням непрямої адресації за регістром *DX*.

Примітка. Варто вказати на те, що апаратна частина не використовує порти від F8*h* до FF*h* для вводу/виводу, оскільки вони зарезервовані для контролю за зовнішнім процесором і для інших можливих розширень процесора в майбутньому.

POP – вибірка слова зі стеку. Прапори не міняються. Команда: *POP destination*. Логіка: *destination* = (*SP*) *SP* = *SP* + 2. Команда *POP* пересилає слово з верхівки стеку за адресою *destination*, потім збільшує покажчик стеку *SP* на 2, щоб він указував на нову верхівку стеку.

Примітка. Як операнд-приймач можна використовувати будь-який 16-розрядний регістр (крім *CS*) або комірку пам'яті.

POPF – пересилання слова зі стеку в регістр *FLAGS*. Прапори міняються: *O*, *D*, *I*, *T*, *S*, *Z*, *A*, *P*, *C*. Команда: *POPF*. Логіка: *flag-register* = (*SP*) *SP* = *SP* + 2. Команда *POPF* пересилає слово з верхівки стеку в регістр *FLAGS*, змінюючи значення всіх ознак, потім збільшує покажчик стеку *SP* на 2, щоб він указував на нову верхівку стеку.

PUSH – завантаження слова в стек. Прапори не міняються. Команда: *PUSH source*. Логіка: *SP* = *SP* – 2 (*SP*) = *source*. Команда *PUSH* зменшує значення покажчика стеку *SP* на 2, потім пересилає операнд у нову верхівку стеку. Операндом *source* не може бути 8-бітний регістр.

Примітка. Навіть якщо *source* вказує на байт, у стек пересилається ціле слово.

PUSHF – завантаження вмісту регістра *FLAGS* у стек. Прапори не міняються. Команда: *PUSHF*. Логіка: *SP* = *SP* – 2 (*SP*) = *flag-register*. Команда *PUSHF* зменшує значення покажчика стеку *SP* на 2, потім пересилає слово з регістра *FLAGS* у верхівку стеку.

SAHF – завантаження регістра *AH* у регістр прапорів. Міняються прапори: *O*, *D*, *I*, *T*, *S*, *Z*, *A*, *P*, *C*. Команда: *SAHF*. Логіка: біти регістра прапорів *FLAGS*: *S*, *Z*, *A*, *P*, *C* відповідають бітам регістра *AH*: 7, 6, 4, 2, 0. Команда *SAHF* копіює біти регістра *AH* з номерами 0, 2, 4, 6 і 7 у регістр *FLAGS*, заміняючи поточні значення ознак переносу, парності, допоміжного переносу, нульового результату та знака.

Примітка. Ця команда використовується, в основному, з метою забезпечення сумісності мікропроцесорів сімей 8080/8085 і 8086. Після виконання цієї команди ознаки переповнення, напрямку, переривання й трасування не змінюються.

XCHG – обмін значеннями. Прапори не міняються. Команда: *XCHG destination*, *source*. Логіка: *destination* ← → *source*. Команда *XCHG* обмінює значення своїх операндів, які можуть бути байтами або словами.

Примітка. Ця команда в парі із префіксом *LOCK* корисна, зокрема, при реалізації семафорів для управління розділеними ресурсами.

XLAT – кодування *AL* за таблицею. Прапори не міняються. Команда: *XLAT translate-table*. Логіка: *AL* = (*BX* + *AL*). Команда *XLAT* переводить байт відповідно до таблиці перетворень. Покажчик 256-байтової таблиці перетворень перебуває в *BX*. Байт, який потрібно перевести, розташований в *AL*. Після виконання команди *XLAT* байт у регістрі *AL* за-

міняється на байт, зміщений на N байтів (де $N = AL$) від початку таблиці перетворень.

Примітка. Таблиця перетворень може містити менше 256 байтів. Операнд, тобто *translate-table*, є необов'язковим, оскільки покажчик таблиці повинен бути завантажений у *BX* ще до початку виконання команди.

Завдання

Із таблиці, що знаходиться в пам'яті за адресою 30*h*, вибрати старшу частину коефіцієнта ділення для каналу 2 таймера (молодша частина = 0). Передати в порт за адресою 43*h* число B6*h*, потім у порт за адресою 42*h* коефіцієнт ділення (обидві частини). Отримати стан порта за адресою 61*h*, зберегти його у стеку, два молодших біти встановити в 1 та передати назад. Створити паузу. Витягти зі стеку дані і передати їх у порт за адресою 61*h*.

На рис. 1.1 наведено рекомендований алгоритм виконання завдання.

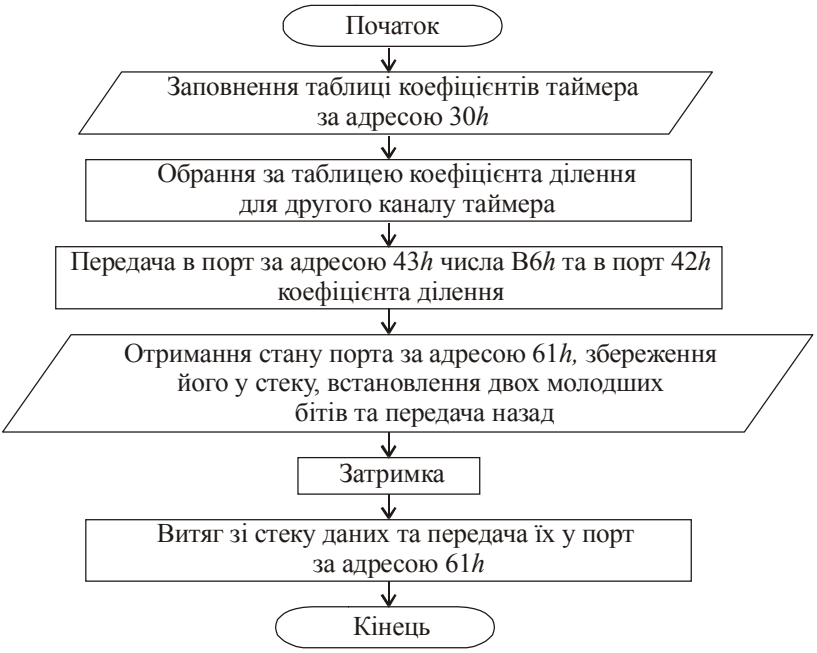


Рис. 1.1. Алгоритм програми роботи з таймером

Запитання до захисту лабораторної роботи

1. Які команди мови асемблера дають можливість працювати з портами вводу/виводу?

2. Описати можливості використання портів вводу/виводу для програмування таймера.

3. Яке переривання викликає системний таймер?

4. Описати процес програмування таймера з використанням портів вводу/виводу.

5. Описати процес зчитування значення лічильника з використанням портів вводу/виводу.

6. Як організувати затримку з використанням функції *BIOS* або з використанням пустого циклу?

7. Що потрібно зробити для включення динаміка ПК?

8. Які припустимі операнди в команді *MOV*?

9. За допомогою якої команди можна здійснити обмін даними між операндами?

10. Як можна зберегти (відновити) вміст регістра прапорів?

11. Як здійснюється завантаження дальніх покажчиків?

12. Як працює команда перетворення даних *XLAT*?

Лабораторна робота № 2

Тема: КОМАНДИ АРИФМЕТИЧНИХ ОПЕРАЦІЙ

Хід роботи

1. Розглянути та вивчити арифметичні команди асемблера.

2. Для закріплення отриманих теоретичних знань створити програму множення двох 32-бітних чисел.

3. Підготуватися до захисту лабораторної роботи за наведеними запитаннями.

Арифметичні команди

За типом використовуваних даних *арифметичні команди* поділяються на команди для роботи з цілими числами та з числами із плаваючою точкою. Команди для роботи із цілими числами у свою чергу поділяються на команди для роботи із двійковими й двійково-десятковими даними. Останній тип даних використовується в системах управління

базами. За довжиною використовуваних даних команди для цілих чисел поділяються на команди для роботи зі словами (2 байти) та байтами. Нижче будуть розглянуті команди для роботи із цілими двійковими числами.

Правила використання арифметичних команд

1. Для цілочислових операндів завжди виходить цілочисловий результат, навіть для операції ділення формується ціла частина частки й остача зі знаком діленого.

2. Довжини операндів повинні збігатися, довжина результату дорівнює довжині операндів.

Виключення 1. При виконанні операції множення добуток може бути у два рази довше співмножників.

Виключення 2. При виконанні операції ділення ділене повинне бути у два рази довше частки. Остача має довжину дільника.

Огляд арифметичних команд

Арифметичні команди поділяються на основні, спеціальні та додаткові. *Основні* арифметичні *команди* задані у табл. 2.1.

Таблиця 2.1

Призначення	Загальний вид	Коментар
Додавання	*add* < Оп1 >, < Оп2 > *xadd* < Оп1 >, < Оп2 >	Оп1 = Оп1 + Оп2 Оп1(Оп2, Оп1) = Оп1 + Оп2
Віднімання	*sub* < Оп1 >, < Оп2 >	Оп1 = Оп1 – Оп2
Множення беззнакове	*mul* < Оп >	$\begin{Bmatrix} AX \\ DX, AX \end{Bmatrix} = \begin{Bmatrix} AL \\ AX \end{Bmatrix} \cdot$ Оп
Множення знакове	*imul* < Оп >	$\begin{Bmatrix} AX \\ DX, AX \end{Bmatrix} = \begin{Bmatrix} AL \\ AX \end{Bmatrix} \cdot$ Оп
Множення знакове двохоперандне	*imul* < Оп1 >, < Оп2 >	Оп1 = Оп1 · Оп2
Множення знакове трьохоперандне	*imul* < Оп1 >, < Оп2 >, константа	Оп1 = Оп2 · константа
Ділення беззнакове	*div* < Оп >	$\begin{Bmatrix} AL \\ AX \end{Bmatrix} = \begin{Bmatrix} AX \\ DX, \ AX \end{Bmatrix} /$ Оп

Продовж. табл. 2.1

Призначення	Загальний вид	Коментар
Ділення знакове	*idiv* < Оп >	$\begin{Bmatrix} AH \\ DX \end{Bmatrix} = \begin{Bmatrix} AX \\ DX,\ AX \end{Bmatrix}$ Оп

У всіх командах не допускається дві адреси пам'яті. При виконанні додавання за допомогою команди *xadd* другий операнд повинен бути регістром. Команди множення із двома-трьома операндами є тільки для знакових множень (множення з урахуванням знака). Ці команди формують добуток довжиною співмножників, тому їх недоцільно використовувати у випадку можливого переповнення. Ділення виконується націло, дробова частина результату відкидається. Одночасно із часткою формується остача від ділення. Якщо при виконанні ділення частка не міститься у відведене для неї поле, виникає особлива ситуація "ділення на нуль". Команда ділення може вимагати спеціальної підготовки для запису діленого. Підготовка полягає в розширенні діленого знаковим розрядом для чисел зі знаком і обнулінні старшої частини діленого для чисел без знака. Для цих цілей можна використовувати команди пересилання або спеціальні команди. Команди пересилання (*MOV*) розглянуті вище.

Спеціальні команди. Ділене завжди повинно бути розташоване у фіксованих регістрах, які залежать від його довжини, тому команди розширення діленого не мають операндів. Код команди визначається типом перетворюваних даних. Команди для підготовки ділення задані у табл. 2.2.

Таблиця 2.2

Виконуване перетворення	Код	Вихідне дане	Результат
Байт у слово	*CBW*	*AL*	*AX*
Слово в подвійне слово	*CWD*	*AX*	*DX, AX*

Число, для завдання якого потрібно більше одного машинного слова, будемо називати числом із багатократною точністю. Такі числа широко використовуються при рішенні різних практичних завдань. Так,

криптографія на відкритих ключах побудована на числах із багатократною точністю, довжина яких 1024 і більше бітів. В основі обчислень із багатократною точністю лежать обчислення "у стовпчик". У цьому випадку на кожному кроці потрібно виконувати операції з однократною точністю. Для обліку можливих переносів у старший розряд при додаванні й позичанні із старшого розряду при відніманні використовуються спеціальні команди, представлені в табл. 2.3.

Таблиця 2.3

Призначення	Загальний вид	Коментар
Додавання	*adc* < Оп1 >, < Оп2 >	Оп1 = Оп1 + Оп2 + біт переносу
Віднімання	*sbc* < Оп1 >, < Оп2 >	Оп1 = Оп1 – Оп2 – біт переносу

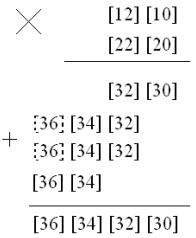
При переповненні чи позичанні із старшого розряду формується біт переносу, що враховується при виконанні операцій додавання й віднімання для чергової частини числа.

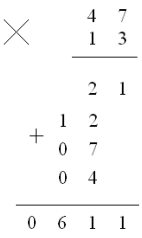
Додаткові команди. Для деяких операцій, таких, як зміна цілого числа на одиницю або його знак, передбачені спеціальні команди, що задані в табл. 2.4. Використання цих команд замість команд загального призначення приводить до генерування більш оптимального коду.

Таблиця 2.4

Призначення	Загальний вид	Коментар
Збільшення на 1	*inc* < Оп >	Оп++
Зменшення на 1	*dec* < Оп >	Оп – –
Інвертування знака	*neg* < Оп >	Оп = – Оп

Завдання





Потрібно помножити два 32-бітні числа. Перше число зберігається в пам'яті за адресою 10*h*, друге – за адресою 20*h*. Результат повинен бути розміщений за адресою 30*h*.

На рис. 2.1 наведено рекомендований алгоритм виконання завдання.

Рис. 2.1. Алгоритм множення двох 32-бітних чисел

Запитання до захисту лабораторної роботи

1. З якими числами можлива робота ПК на базі процесора *I*8086? З якими числами можлива робота ПК на базі сучасних процесорів?

2. Які арифметичні команди використовуються для додавання операндів?

3. Які арифметичні команди використовуються для віднімання операндів?

4. Які можливості має асемблер для знакових і беззнакових множень та ділень?

5. Яким чином можна робити операції із числами, розрядність яких перевищує розрядність процесора?

6. Число якої розрядності отримаємо в результаті множення двох 32-бітних чисел?

7. Який алгоритм дозволяє множення 32-бітних операндів?

8. Яким чином при арифметичних операціях ураховується перенос у старший розряд чи позичання із старшого розряду?

9. Чи можна поперед команди, яка використовує перенос у старший розряд або позичання із старшого розряду, вставляти команди, які можуть змінити регістр прапорів?

Лабораторна робота № 3

Тема: КОМАНДИ ЛОГІЧНИХ ОПЕРАЦІЙ ТА КОМАНДИ ЗМІЩЕНЬ

Хід роботи

1. Розглянути логічні команди, команди зміщень, команди переходів.

2. На основі вивченого матеріалу створити кодуючий та декодуючий пристрій, стійкий до спотворення сигналу.

3. Підготуватися до захисту лабораторної роботи за наведеними запитаннями.

Робота з бітами

Основні команди для роботи з бітами задані у табл. 3.1.

Таблиця 3.1

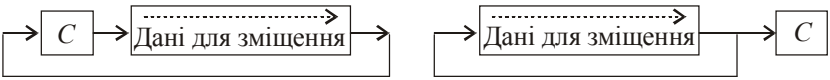
Номер	Призначення	Код	Виконувані дії	Формовані прапори
1	Побітове додавання	*OR*	Оп1& = Оп2	$C = 0$, $O = 0$, Z, P, S
2	Побітове множення	*AND*	Оп1& = Оп2	$C = 0$, $O = 0$, Z, P, S

Продовж. табл. 3.1

Номер	Призначення	Код	Виконувані дії	Формовані прапори
3	Перевірка	*TEST*	Оп1& Оп2	*C* = 0, *O* = 0, *Z*, *P*, *S*
4	Побітове заперечення	*NOT*	~ Оп1	Прапори не змінюються
5	Додавання за модулем 2	*XOR*	Оп1^ = Оп2	*C* = 0, *O* =0, *Z*, *P*, *S*

Команди зміщень

Зміщення може бути *звичайним* і *циклічним*. При звичайному зміщенні розряди, що виходять за розрядну сітку, знищуються. Для циклічного зміщення вони записуються на місце звільнених розрядів. Звичайне зміщення може бути арифметичним і логічним. При арифметичному зміщенні вправо вільні позиції ліворуч заповнюються знаковим (старшим) розрядом. При логічному зміщенні вправо вільні розряди ліворуч заповнюються нулями. Арифметичне зміщення вправо використовується зазвичай для поділу націло знакового числа на 2^n, а логічне – для беззнакового числа. Операції зміщень уліво для обох типів команд виконуються однаково. Циклічне зміщення може бути з переносом і без переносу. Схема виконання команд для обох типів зміщень на 1 біт вправо показана на рис. 3.1.



а ***б***

Рис. 3.1. Схема виконання циклічного зміщення:
а – циклічне зміщення з переносом; ***б*** – циклічне зміщення без переносу

Висунутий біт завжди вміщається у біт переносу *C*. При циклічному зміщенні попередній вміст біта переносу записується у звільнений розряд, при звичайному – губиться, а у звільнений розряд записується висунутий біт. При будь-якому зміщенні останній із висунутих розрядів записується замість біта переносу.

Кодування команд зміщень. Перша буква коду визначає тип зміщення. Буква *S* відповідає звичайному, а буква *R* циклічному зміщенню. Друга буква уточнює тип зміщення. При звичайному зміщенні використовується буква *A* для арифметичного й буква *H* для логічного зміщень. При циклічному зміщенні друга буква *C* відповідає зміщенню з переносом і буква *O* зміщенню без переносу. Третя буква визначає напрямок зміщення. Буква *R* відповідає зміщенню вправо,

буква *L* зміщенню вліво. Таким чином, коди команд зміщень мають вигляд $\begin{Bmatrix} S \\ R \end{Bmatrix}$ $\begin{Bmatrix} H \\ A \\ O \\ C \end{Bmatrix}$ $\begin{Bmatrix} L \\ R \end{Bmatrix}$.

Загальний вид команд зміщень: код Оп1, Оп2, Оп1 – дані для зміщення, які можуть бути байтом або словом і задаються в загальному регістрі або пам'яті, Оп2 – задає, на скільки розрядів виконується зміщення, і може бути константою або регістром *CL*, у який попередньо необхідно записати константу зміщення.

На рис. 3.2 (див. с. 16) наведено рекомендований алгоритм виконання завдання.

Запитання до захисту лабораторної роботи

1. Які логічні команди для роботи з бітами ви знаєте?
2. Навести алгоритм правого та лівого циклічного зміщень.
3. Навести алгоритм арифметичного та логічного зміщень.
4. Чим відрізняється циклічне зміщення від звичайного?
5. Які саме прапори містить регістр прапорів?
6. Які команди можуть змінювати значення прапорів?
7. Які команди умовного переходу ви знаєте?
8. Навести алгоритм кодування за кодом Хемінга.
9. Навести алгоритм декодування за кодом Хемінга.

Лабораторна робота № 4

Тема: КОМАНДИ ПЕРЕДАЧІ УПРАВЛІННЯ

Хід роботи

1. Розглянути команди передачі управління.

2. На основі вивченого матеріалу створити програму сортування масиву за зростанням та убуванням. Варіанти створення програми: використання налагоджувального середовища; використання тегів *Pascal asm ... end*; написання повноцінної програми мовою асемблера.

3. Підготуватися до захисту лабораторної роботи за наведеними запитаннями.

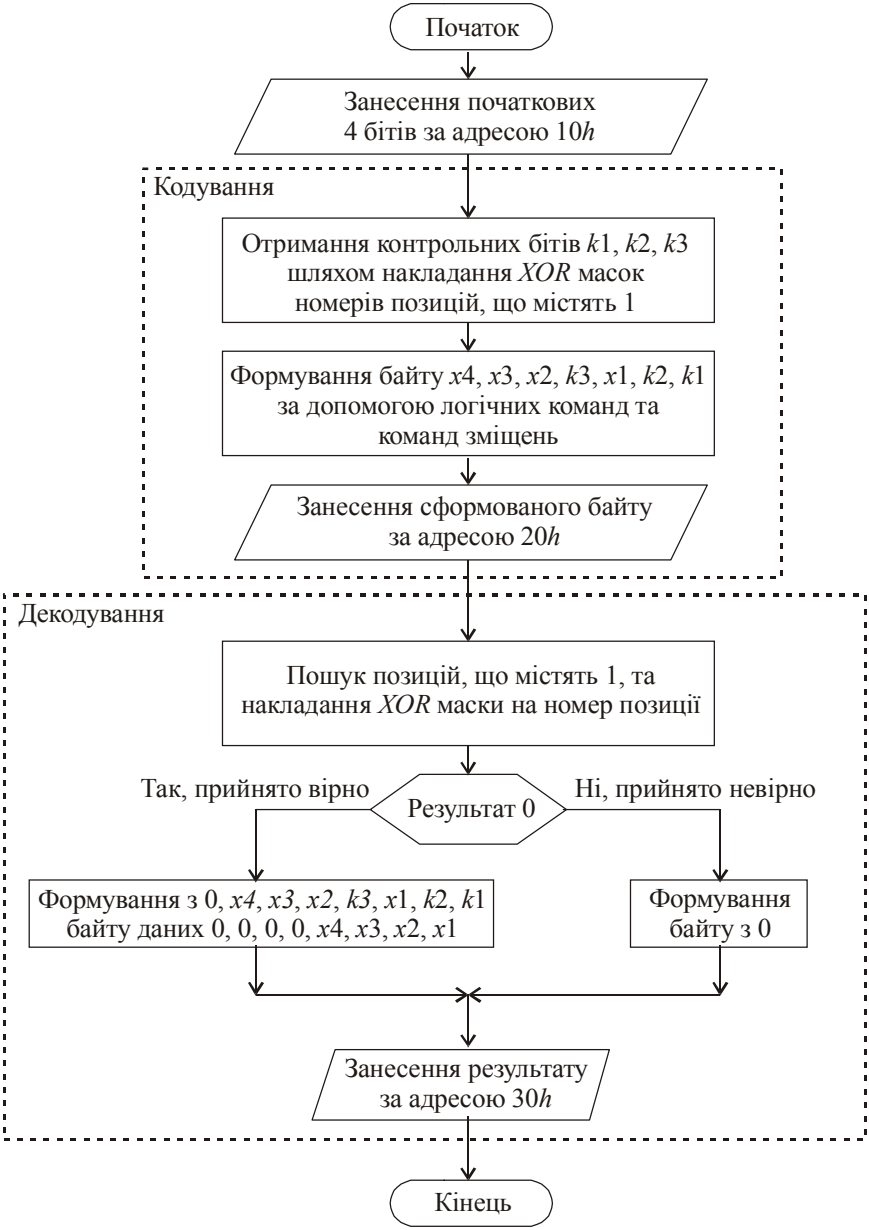


Рис. 3.2. Алгоритм Хемінга

Організація розгалужень і циклів

Команди безумовного переходу поділяються на команди прямого переходу й команди непрямого переходу, на команди короткі й довгі. Команди прямого переходу вимагають, щоб у команді переходу стояла мітка, визначена в даному модулі. Команди непрямого переходу як операнд приймають адресу з адресою переходу. У командах короткого переходу для завдання зміщення, що відповідає мітці, використовується один байт, у противному випадку – чотири. Загальний вид команд безумовного переходу: *JMP* [*SHORT*] операнд.

Команди умовного переходу. Арифметичні команди формують коди умов залежно від результату виконання команди: перенос *C* – якщо є перенос за межі розрядної сітки комірки; парність *P* – якщо кількість бітів молодшого байта результату парна; знак *S* – результат негативний; нуль *Z* – результат дорівнює нулю; *O* – переповнення.

Ознака результату записується в регістр прапорів. Регістр прапорів – 16 (32)-бітне число, в якому кожний прапор записується у фіксований біт. Для розглянутих вище прапорів використовуються біти: *C* – біт 0; *P* – біт 2; *Z* – біт 6 ; *S* – біт 7; *O* – біт 11. Вміст регістра прапорів може бути записано в стек (команда *pushfd*) і прочитано зі стеку (наприклад, команда *pop eax*). Спеціально для зміни бітів регістра прапорів використовуються команди:

команда порівняння: *CMP* Оп1, Оп2. Операнди задаються так, як для арифметичних команд. Фактично виконується команда віднімання, але результат не записується замість першого даного, а тільки формується регістр прапорів;

команда *TEST* Оп1, Оп2. Операнди задаються так, як для арифметичних команд. Фактично виконується операція порозрядного множення, але результат не записується в Оп1, а формуються прапори. Команда використовується для порівняння з нулем усього числа або окремих його бітів;

команда *BT* Оп, *константа*. Команда перевіряє вміст біта, номер якого заданий константою в операнді. Результат записується в біт переносу регістра прапорів.

Команди непрямого умовного переходу. Загальний вид команди: [Мітка 1] Код Мітка 2.

Непрямий умовний перехід не використовується. При трансляції команди замість мітки записується різниця між адресою мітки й адресою

команди, що знаходиться після команди умовного переходу. Якщо ця різниця міститься в один байт (–128...127), команда називається короткою (*short*). Для команд із посиланням назад компілятор сам визначає тип команди. Для посилань назад тип команди задається програмістом, а якщо він не заданий, то приймається звичайним. Якщо програміст задав короткий перехід, а він неможливий, компілятор виводить повідомлення: *RELATIVE JUMP OUT OF RANGE*. Команди переходу поділяються на переходи за прапорами, переходи для знакових і беззнакових даних (табл. 4.1).

Таблиця 4.1

Переходи за прапорами		Переходи для знакових даних		Переходи для беззнакових даних	
Прямий	Зворотній	Прямий	Зворотній	Прямий	Зворотній
JC	*JNC*	*JL* (<)	*JNL* (>=)	*JB* (<)	*JNB* (>=)
JP	*JNP*	*JLE* (<=)	*JNLE* (>)	*JBE* (<=)	*JNBE* (>)
JZ	*JNZ*	*JE* (==)	*JNE* (!=)	*JE* (==)	*JNE* (!=)
JS	*JNS*	*JG* (>)	*JNG* (<=)	*JA* (>)	*JNA* (<=)
JO	*JNO*	*JGE* (>=)	*JNGE* (<)	*JAE* (>=)	*JNAE* (<)

У табл. 4.2 подані зведені команди передачі управління.

Таблиця 4.2

Код команди	Опис команди
JMP target 16 *JMP NEAR taget 8 JMP reg* *JMP mem*	Внутрішньосегментний безумовний перехід до цільової адреси *target* (перехід у межах сегмента довжиною 64 Кбайт)
JMP FAR target JMP FAR mem	Міжсегментний безумовний перехід до цільової адреси *target* (перехід у межах усієї пам'яті довжиною 1 Мбайт)
JCX target	Перехід, якщо $CX = 0$
LOOP target	Циклічний перехід, якщо $CX \neq 0$
LOOPE (LOOPZ) target	Циклічний перехід, якщо $CX \neq 0$, а $ZF = 1$
LOOPNE (LOOPNZ) target	Циклічний перехід, якщо $CX \neq 0$, а $ZF = 0$
JA (JNBE) target	Перехід, якщо перший беззнаковий операнд більше за другий ($CF = ZF = 0$)
JAE (JNB) target	Перехід, якщо перше беззнакове число не менше за друге

Продовж. табл. 4.2

Код команди	Опис команди
JB (JC) target	Перехід, якщо перше беззнакове число менше за друге (*CF* = 1)
JE (JZ)	Перехід, якщо числа рівні (*ZF* = 1)
JG (JNLE) target	Перехід, якщо перше знакове число більше за друге (*CF* = (*ZF* & *OF*))
JGE (JNL) target	Перехід, якщо перше знакове число більше за друге (*SF* = *OF*)
JL (JNGE) target	Перехід, якщо перше знакове число менше за друге (*SF* ≠ *OF*)
JLE (JNG) target	Перехід, якщо перше знакове число менше або дорівнює другому (*SF* ≠ *OF* або *ZF* = 0)
JNC(JAE/JNB) target	Перехід, якщо немає переносу (*CF* = 0)
JNE (JNZ) target	Перехід, якщо числа не рівні (*ZF* = 0)
CALL NEAR target CALL NEAR reg CALL NEAR mem	Внутрішньосегментний виклик процедури. Виклик у межах сегмента довжиною 64 Кбайт
CALL FAR target CALL FAR mem	Сегментний виклик процедури (виклик до границі всієї пам'яті довжиною 1 Мбайт)
RET RET NEAR RET(n) RET NEAR (n)	Повернення із внутрішньосегментної процедури. Не обов'язковий параметр *n* задає корекцію значення покажчика стеку
RET FAR RET FAR(n)	Повернення з міжсегментної процедури

Завдання

Є масив з *m* байтів (одномірний). За адресою 10*h* є довжина масиву *N*. За адресою 11*h* буде перебувати байт, що буде вказувати напрямок. Якщо цей байт дорівнює 0, то треба сортувати масив за зростанням, якщо не дорівнює 0, то за убуванням. Необхідно одержати середнє арифметичне елементів масиву. При цьому потрібно використовувати команди умовних переходів, команди циклів і команди підпрограм. За адресою 12*h* заноситься ціла частина, за адресою 13*h* – чисельник дробового числа, а за адресою 14*h* – знаменник.

На рис. 4.1 наведено рекомендований алгоритм виконання завдання.

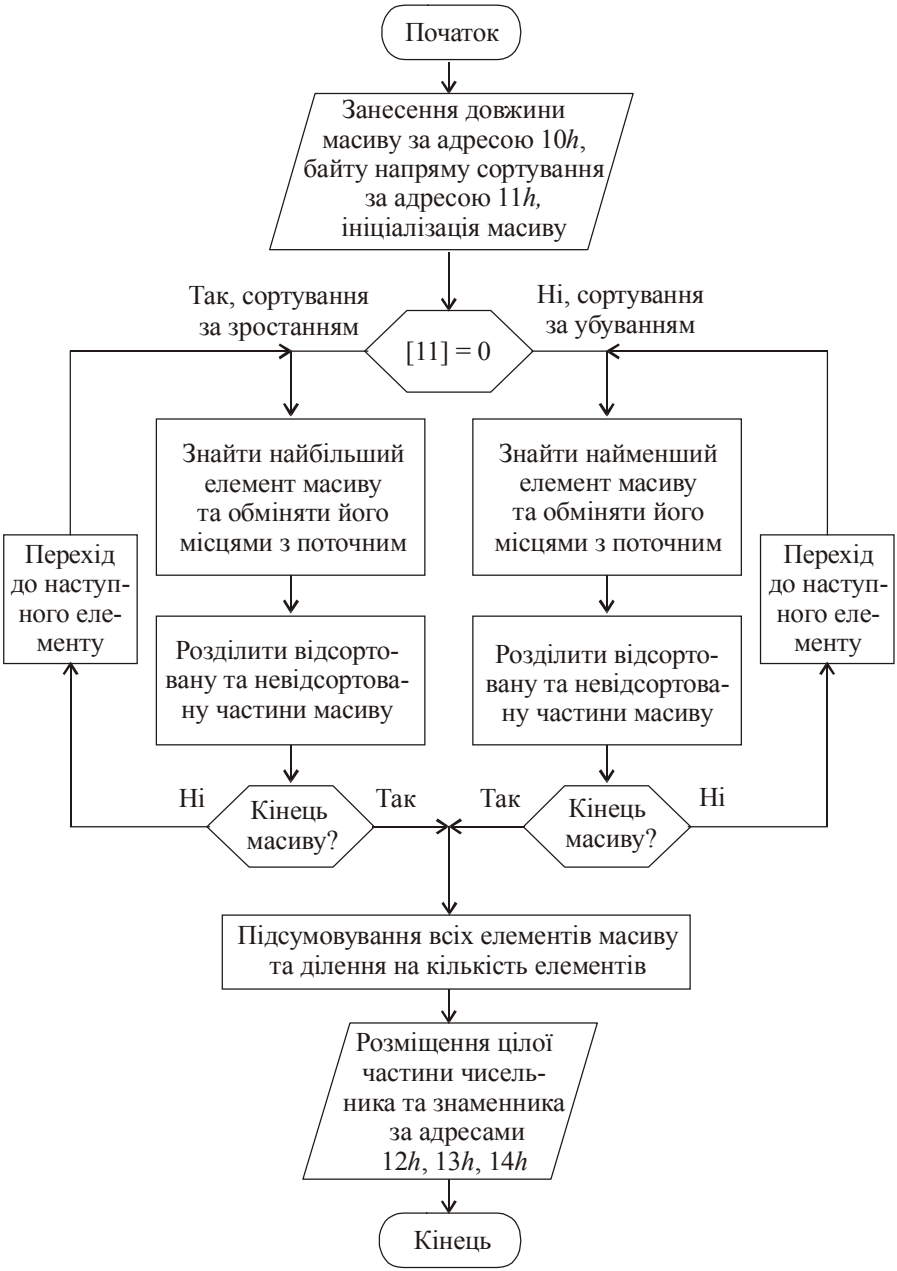


Рис. 4.1. Алгоритм сортування масиву

Запитання до захисту лабораторної роботи

1. Яке загальне застосування команд передачі даних?
2. Які команди передачі даних ви знаєте?
3. Які існують види переходів?
4. Які різновиди безумовного переходу ви знаєте?
5. Перерахувати відомі вам види сортування.
6. Навести алгоритм сортування, який рекомендовано для виконання цієї роботи.
7. Яким чином команди умовних переходів отримують "інформацію" про результат попередніх команд?
8. Які прапори використовуються для команд умовних переходів? Навести варіанти їх можливих співвідношень.
9. Яка команда передачі даних мови асемблера є еквівалентною до процедур та функцій мов високого рівня?
10. Які команди передачі даних можуть бути використані для організації циклів?

Лабораторна робота № 5

Тема: КОМАНДИ ДЛЯ ОБРОБКИ БЛОКІВ ДАНИХ

Хід роботи

1. Розглянути команди обробки блоків даних.
2. На основі вивченого матеріалу виконати завдання із пошуку тексту.
3. Підготуватися до захисту лабораторної роботи за наведеними запитаннями.

Теоретичні відомості

Під ланцюжком розуміється послідовність будь-яких контекстно-зв'язаних байтів або слів, що перебувають у суміжних комірках пам'яті. Ланцюжок-джерело за замовчуванням перебуває в сегменті даних *DS*, ланцюжок-одержувач – у додатковому сегменті *ES*. Регістр *SI* містить зміщення поточного елемента в ланцюжку-джерелі від початку сегмента даних. А регістр *DI* містить зміщення поточного елемента ланцюжка-одержувача відносно початку додаткового сегмента. При виконанні ланцюгових команд регістри *SI* і *DI* автоматично модифікуються, щоб адресувати наступні елементи ланцюжка. Прапорець *DF* визначає на-

прямок обробки ланцюжків. Якщо він дорівнює 0, то здійснюється автоінкремент, якщо – 1 – автодекремент регістрів *SI* і *DI*.

Команда передачі ланцюжка: інструкція *MOVS* копіює байт слово за адресою [*SI*] у байт або слово в *ES*:[*DI*]. Операнд-одержувач повинен бути адресованим через регістр *ES*, перевизначення сегмента в цільовому операнді не допускається. У операнді-джерелі перевизначення сегмента використовувати можна. За замовчуванням використовується регістр *DS*. Адреси вихідного й цільового операнда визначаються винятково вмістом регістрів *SI* і *DI*. Перед виконанням інструкції *MOVS* у *SI* і *DI* завантажуються коректні значення індексу. Синонімом інструкції *MOVS* для роботи з байтами є інструкція *MOVSB*. Після переміщення даних виконується автоматичне просування регістрів *SI* і *DI*. Якщо прапор напрямку дорівнює 0 (була виконана інструкція *CLD*), то регістри інкрементуються, якщо прапор напрямку дорівнює 1 (була виконана інструкція *STD*), то декрементуються. При завантаженні байта регістри збільшуються або зменшуються на 1, при завантаженні слова – на 2. Інструкції *MOVS* може передувати префікс *REP*. Це робиться для переміщення блока з *N* байтів або слів (де $N = CX$).

Команда завантаження елемента ланцюжка в акумулятор: інструкція *LODS* завантажує регістр *AL* або *AX* байтом або словом з пам'яті за адресою, що вказує вихідний індексний регістр. Після виконання пересилання вихідний індексний регістр автоматично просувається (збільшується або зменшується). Якщо прапорець напрямку дорівнює 0 (була виконана інструкція *CLD*), то індекс джерела збільшується, якщо прапорець напрямку дорівнює 1 (була виконана інструкція *STD*), то він зменшується. При завантаженні байта індекс збільшується або зменшується на 1, при завантаженні слова – на 2. Як індексний регістр джерела використовується регістр *SI*. Адреса вихідних даних визначається винятково вмістом *SI*. Перед виконанням інструкції *LODS* потрібно завантажити в регістр *SI* коректне значення індексу. Синонімом для інструкції *LODS* при роботі зі словом є інструкція *LODSW*. Інструкції *LODS* може передувати префікс *REP*. Однак частіше *LODS* використовується в конструкції *LOOP*, оскільки зазвичай потрібна подальша обробка даних, завантажених у регістр *AX* або *AL*.

Команда порівняння: *CMPSB CMPSW*.

Команда сканування ланцюжків: *SCASB SCASW*, де шуканий елемент заноситься в акумулятор.

Команда запам'ятовування вмісту ланцюжка в акумулятор: *STOSB STOSW* з акумулятора в ланцюжок-одержувач *DI*.

Префікси повторень: *REP* – повторювати доки *CX* не дорівнюватиме 0; *REPE* – повторювати доки *CX* не дорівнюватиме 0 і *ZF* = 1 *REPZ*; *REPNE* – повторювати доки *CX* не дорівнюватиме 0 і *ZF* = 0 *REPNZ*.

Завдання

За адресою 10*h* перебуває текст. Ознакою закінчення тексту є 0. За адресою 00...07*h* перебуває шуканий текст, що також завершується кодом 0.

Потрібно знайти в тексті 2, що розміщений за адресою 10*h*, шуканий текст. Якщо текст знайдений у повному обсязі, то за адресою 50*h* заносимо повідомлення, що перебуває за адресою 30*h*, у противному випадку за адресою 50*h* заносимо повідомлення, що починається за адресою 40*h*.

На рис. 5.1 наведено рекомендований алгоритм виконання завдання.

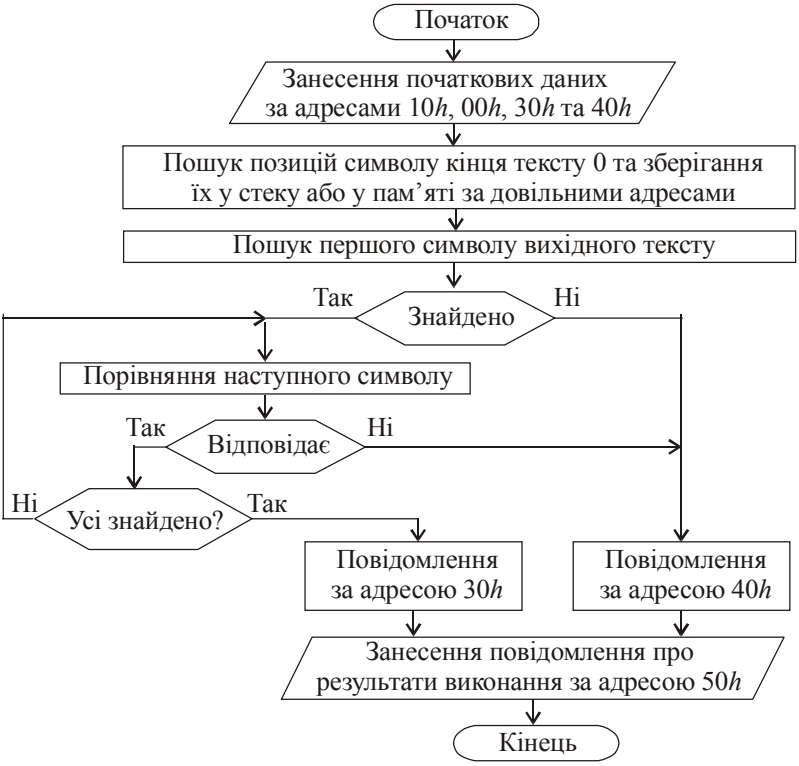


Рис. 5.1. Алгоритм пошуку послідовності символів у тексті

Запитання до захисту лабораторної роботи

1. Навіщо потрібні команди обробки блоків даних? Їх практичне застосування і загальна схема роботи.

2. Які команди для обробки блоків даних ви знаєте? Навести за групами.

3. Яким чином для команд обробки блоків даних застосовується прапорець *DF*?

4. Як розрізняється синтаксис команд обробки блоків даних у залежності від розміру блока?

5. Що таке префікси? Їх застосування.

6. Чим відрізняються команди порівняння *CMPS* від *CMP*?

7. Навести алгоритм пошуку ланки даних у певному тексті.

8. Який регістр використовується для зберігання кількості умовних блоків для команд обробки блоків?

9. Навести адресацію ланок джерела та одержувача.

ДОДАТКИ

Список умовних позначень

src – операнд-джерело;
dst – операнд-призначення;
reg – 8(16)-розрядний регістр загального призначення;
sr – сегментний регістр;
mem – 8(16)-розрядна комірка пам'яті;
immed – безпосередній операнд;
disp – 8(16)-розрядне зміщення при визначенні адреси;
target – мітка, до якої здійснюється перехід;
seg target – перша логічна адреса (сегментна адреса) мітки *target*;
offset target – друга логічна адреса (зміщення у сегменті) мітки *target*;
A – акумулятор *AL* або *AX*;
m[*disp*] – комірка пам'яті за ефективною адресою *EA* = *disp*.

У таблицях подані призначення прапорів (табл. 1) та система команд мікропроцесора *I*8086 (табл. 2).

Таблиця 1

Позначення	Призначення	Розрядність	
		8	16
AF	*Auxiliary Flag* – прапорець допоміжного перенесення/позики з молодшої тетради в старшу (з розряду *D*3 в розряд *D*4). Використовується при десятковій арифметиці	+	–
CF	*Carry Flag* – прапорець перенесення/позики. Встановлюється при виході результату додавання (віднімання) беззнакових операндів за межу діапазону. У командах зміщення прапорець *CF* фіксує значення старшого біта	+	+
OF	*Overflow Flag* – прапорець переповнення, встановлюється при виході знакового результату за межі діапазону	+	+
SF	*Sign Flag* – прапорець знака. Дублює значення старшого біта результату. *SF* = 0 для позитивних чисел і *SF* = 1 – для негативних	+	+
PF	*Parity Flag* – прапорець паритету (парності). Встановлюється при парному числі одиниць у результаті	+	–
ZF	*Zero Flag* – прапорець нульового результату. Встановлюється при отриманні нульового результату операції	+	+
DF	*Direction Flag* – прапорець управління напрямом у рядкових операціях. При *DF* = 1 індексні регістри *SI*, *DI*, що беруть участь у рядкових операціях, автоматично декрементуються на кількість байтів операнда, при *DF* = 0 – інкрементуються		
IF	*Interrupt-enable Flag* – прапорець дозволу переривань. При *IF* = 1 дозволяється виконання маскованих апаратних переривань		
TF	*Trap Flag* – прапорець трасування (покрокового режиму). При його встановленні після виконання кожної команди викликається внутрішнє переривання 1 (INT 1)		

Таблиця 2

Мнемокод команди	Опис команди	Алгоритм команди	Зміна прапора
КОМАНДИ ПЕРЕДАЧІ ІНФОРМАЦІЇ			
КОМАНДИ ПЕРЕСИЛАННЯ			
MOV dst, src	Пересилання даних із регістра, комірки пам'яті або операнда у регістр або пам'ять	*dst*←*src*	Не діє
XCHG dst, src	Обмін даними між регістрами або регістром і пам'яттю	*dst*←→*src*	-//-
XLAT	Перекодування вмісту *AL* у значення байта пам'яті за адресою *ES*:[*BX* + (*AL*)]	*AL*←*ES*:[*BX*+(*AL*)]	-//-
LEA dst, mem	Завантаження ефективної адреси комірки пам'яті у регістр	*reg*←*EA*	-//-
LDS dst, mem	Завантаження в регістр слова із комірки пам'яті, у *DS* – наступного слова з наступної комірки	*dst*← [*mem*]; *DS* ← [*mem*+2]	-//-
LES dst, mem	Завантаження в регістр слова із комірки пам'яті за адресою [*mem*], в *ES* – наступного слова з комірки за адресою [*mem* + 2]	*dst* ← [*mem*]; *ES* ← [*mem* + 2]	-//-
LAHF	Завантаження молодшого байта регістра прапорів *FL* в *AH*	*AH* ← *FL*	-//-
SAHF	Збереження *AH* у молодшому байті регістра прапорів *FL*	*FL* ← *AH*	*CF*, *AF*, *ZF*, *PF*, *SF*
КОМАНДИ РОБОТИ ЗІ СТЕКОМ			
PUSH src	Пересилання слова з регістра або з пам'яті у стек	[*SS*:*SP*]← *src*	Не діє
PUSHF	Пересилання у стек вмісту регістра прапорів	[*SS*:*SP*]← *F*	-//-
POP dst	Пересилання слова даних зі стеку у регістр або пам'ять	*dst* ← [*SS*:*SP*]	-//-

Продовження табл. 2

Мнемокод команди	Опис команди	Алгоритм команди	Зміна прапора
POPF	Пересилання даних зі стеку у регістр прапорів	$F \leftarrow [SS{:}SP]$	усі
КОМАНДИ ВВОДУ/ВИВОДУ			
IN A, *P*8 (*DX*)	Ввід в акумулятор *A* байта з 8 (16)-розрядного порта за адресою *P*8 або *DX*	*A* ← Порт (*P*8); *A* ← Порт (*DX*)	Не діє
*OUT P*8(*DX*), *A*	Вивід байта з акумулятора *A* у 8 (16)-розрядний порт за адресою *P*8 або *DX*	Порт (*P*8) ← *A*; Порт (*DX*) ← *A*	Не діє
КОМАНДИ ОБРОБКИ ІНФОРМАЦІЇ			
АРИФМЕТИЧНІ КОМАНДИ			
ADD dst, *src*	Додавання двох операндів	*Dst* = *dst* + *src*	*OF*, *CF*, *AF*, *SF*, *ZF*, *PF*
ADC dst, *src*	Додавання операндів і прапорця *CF* попередньої операції	*Dst* = *dst* + *src* + *CF*	-//-
SUB dst, src	Віднімання операндів	*Dst* = *dst* – *src*	-//-
SBB dst, src	Віднімання байта та прапорця позики *CF* попередньої операції	*Dst* = *dst* – *src* – *CF*	-//-
NEG src	Зміна знака операнда	*Src* = – *src*	-//-
CMP dst, src	Порівняння двох операндів. Регістр прапорів *F* за результатом віднімання	*Dst* – *src*	-//-
INC src	Інкремент (додавання з одиницею)	*Src* = *src* + 1	*OF*, *AF*, *SF*, *ZF*, *PF*
DEC src	Декремент (віднімання одиниці)	*Src* = *src* – 1	-//-
MUL src	Множення *A* на беззнакове значення *src*	*DX*, *AX* ← |*AX*|·|*src*|	*OF*, *CF*; не визначені: *AF*, *ZF*, *SF*, *PF*
IMUL src	Множення *A* на знакове значення *src*	*DX*, *AX* ← *AX* · *src*	-//-

Продовження табл. 2

Мнемокод команди	Опис команди	Алгоритм команди	Зміна прапора
DIV src	Ділення акумулятора на беззнакове число (ділення на нуль викликає переривання *INT* 0)	$\|AX\|:\|src\| \to AL$ (*остача AH*); $\|DX, AX\|:\|src\| \to AX$ (*остача DX*)	Не визначені *OF*, *CF*, *AF*, *ZF*, *SF*, *PF*
IDIV src	Ділення акумулятора на ціле число (ділення на нуль викликає переривання *INT* 0)	$AX : src \to AL$ (остача *AH*); $DX, AX : src \to AX$ (*остача DX*)	-//-
ЛОГІЧНІ КОМАНДИ			
NOT src	Інверсія всіх бітів операнда	$src \leftarrow \overline{src}$	Не діє
AND dst, src	Логічне І двох операндів	$dst \leftarrow dst \wedge src$	*OF*, *SF*, *ZF*, *PF* *CF* = 0; не визначений *AF*
OR dst, src	Логічне АБО двох операндів	$dst \leftarrow dst \vee src$	-//-
XOR dst, src	Логічне виключне АБО двох операндів	$dst \leftarrow dst \oplus src$	-//-
TEST dst, src	Перевірка (логічне І без запису результату)	$dst \wedge src$	-//-
КОМАНДИ ЗМІЩЕННЯ			
RCL/RCR dst, 1 або *dst, CL*	Циклічне зміщення ліворуч/праворуч через біт *CF* на одну або *CL* позиції	$\to dst \to CF \to$; $\leftarrow CF \leftarrow dst \leftarrow$	*OF*, *CF*; не визначений *AF*
SHR dst, 1 *CL*	Зміщення логічне праворуч на одну позицію або *CL* позицій	$0 \to dst \to CF$	*OF*, *CF*, *SF*, *ZF*, *PF*; не визначений *AF*
SAL/SAR dst, 1 або *dst,CL*	Зміщення арифметичне ліворуч/праворуч на одну або *CL* позиції	$CF \leftarrow dst \leftarrow 0$; $dst \leftrightarrow dst \to CF$	*OF* = 0; *CF*, *SF*, *ZF*, *PF*; не визначений *AF*

Продовження табл. 2

Мнемокод команди	Опис команди	Алгоритм команди	Зміна прапора
ROL/ROR dst, 1 або *Dst,CL*	Циклічне зміщення ліворуч /праворуч на одну або *CL* позиції	$CF \leftarrow dst \leftrightarrow dst$; $dst \leftrightarrow dst \rightarrow CF$	*CF*; не визначені *OF*, *AF*
РЯДКОВІ КОМАНДИ			
REP	Префікс повторення рядкових операцій до $CX = 0$	–	Не діє
REPE (*REPZ*)	Префікс умовного повторення при $ZF = 1$, або до обнуління *CF*	–	-//-
REPNE (*REPNZ*)	Префікс умовного повторення при $ZF = 0$, або до обнуління *CF*	–	-//-
MOVSB	Копіювання байта з комірки пам'яті за адресою *DS*: [*SI*] в комірку *ES*: [*DI*]	*ES*:[*DI*] ← *DS*:[*SI*]; $SI \leftarrow SI \pm 1$; $DI \leftarrow DI \pm 1$	-//-
MOVSW	Копіювання слова з *DS*: [*SI*] в *ES*: [*DI*]	*ES*:[*DI*] ← *DS*:[*SI*]; $SI \leftarrow SI \pm 2$; $DI \leftarrow DI \pm 2$	-//-
LODSB	Копіювання байта з *DS*: [*SI*] в *AL*	*AL* ← *DS*:[*SI*]; $SI \leftarrow SI \pm 1$	-//-
LODSW	Копіювання слова з *DS*: [*SI*] в *AX*	*AX* ← *DS*:[*SI*]; $SI \leftarrow SI \pm 2$	-//-
STOSB	Запис байта з *AL* в *ES*: [*DI*]	*ES*:[*DI*]← *AL*; $DI \leftarrow DI \pm 1$	-//-
STOSW	Запис слова з *AX* в *ES*: [*DI*]	*ES*:[*DI*] ← *AX*; $DI \leftarrow DI \pm 2$	-//-
CMPSB (*CMPSW*)	Порівняння байтів *DS*: [*SI*] і *ES*: [*DI*] із записом результату порівняння у регістр прапорів	*ES*:[*DI*] – *DS*:[*SI*]; $SI \leftarrow SI \pm 1(2)$; $DI \leftarrow DI \pm 1(2)$	*OF*, *CF*, *AF*, *SF*, *ZF*, *PF*
SCASB (*SCASW*)	Порівняння байта *DS*: [*SI*] і *AL* (*AX*) із записом результату порівняння у регістр прапорів	*DS*: [*SI*] – *AL*(*AX*); $S \leftarrow SI \pm 1(2)$	-//-

Продовження табл. 2

Мнемокод команди	Опис команди	Алгоритм команди	Зміна прапора
КОМАНДИ ПЕРЕДАЧІ КЕРУВАННЯ			
JMP (*NEAR*) *dst*	Внутрішньосегментний безумовний перехід до цільової адреси *dst* (у сегменті 64 Кбайт)	$IP \leftarrow IP + dst$	Не діє
JMP FAR dst	Міжсегментний безумовний перехід до цільової адреси *dst* (у межах пам'яті 1 Мбайт)	$IP \leftarrow offset\ dst$; $CS \leftarrow seg\ dst$	-//-
JCX dst	Перехід, якщо $CX = 0$	$IP \leftarrow IP + dst$	-//-
LOOP dst	Цикл і перехід, якщо $CX \neq 0$	$CX \leftarrow CX - 1$; $IP \leftarrow IP + dst$	-//-
LOOPE (*LOOPZ*) *dst*	Цикл і перехід, якщо $CX \neq 0$ і $ZF = 1$	$CX \leftarrow CX - 1$; $IP \leftarrow IP + dst$	-//-
LOOPNE (*LOOPNZ*) *dst*	Цикл і перехід, якщо $CX \neq 0$ і $ZF = 0$	$CX \leftarrow CX - 1$; $IP \leftarrow IP + dst$	-//-
JA (*JNBE*) *dst*	Перехід, якщо перший беззнаковий операнд більше за другий ($CF = ZF = 0$)	$IP \leftarrow IP + dst$	-//-
JAE (*JNB*) *dst*	Перехід, якщо перше беззнакове число не менше за друге ($CF = 0$)	$IP \leftarrow IP + dst$	-//-
JB (*JC*) *dst*	Перехід, якщо перше беззнакове число менше за друге ($CF = 1$)	$IP \leftarrow IP + dst$	-//-
JE (*JZ*) *dst*	Перехід, якщо числа дорівнюють 0 ($ZF = 1$)	$IP \leftarrow IP + dst$	-//-
JG (*JNLE*) *dst*	Перехід, якщо перше число більше за друге знакове число ($SF = (ZF \& OF)$)	$IP \leftarrow IP + dst$	-//-
JGE (*JNL*) *dst*	Перехід, якщо перше знакове число не менше за друге ($SF = OF$)	$IP \leftarrow IP + dst$	-//-
JL (*JNGE*) *dst*	Перехід, якщо перше знакове число менше за друге ($SF \neq OF$)	$IP \leftarrow IP + dst$	-//-

Продовження табл. 2

Мнемокод команди	Опис команди	Алгоритм команди	Зміна прапора
JLE (*JNG*) *dst*	Перехід, якщо перше знакове число не більше за друге (*SF* ≠ 0 або *ZF* = 0)	*IP* ← *IP* + *dst*	-//-
JNC (*JAE/JNB*) *dst*	Перехід, якщо немає переносу (*CF* = 0)	*IP* ← *IP* + *dst*	-//-
JNE (*JNZ*) *dst*	Перехід, якщо числа не рівні (*ZF* = 0)	*IP* ← *IP* + *dst*	-//-
JNO dst	Перехід, якщо немає переповнення (*OF* = 0)	*IP* ← *IP* + *dst*	-//-
JNP (*JPO*) *dst*	Перехід, якщо паритет непарний (*PF* = 0)	*IP* ← *IP* + *dst*	-//-
JNS dst	Перехід, якщо результат позитивний (*SF* =0)	*IP* ← *IP* + *dst*	-//-
JO dst	Перехід, якщо є переповнення (*OF* = 1)	*IP* ← *IP* + *dst*	-//-
JP (*JPE*) *dst*	Перехід, якщо паритет парний (*PF* = 1)	*IP* ← *IP* + *dst*	-//-
JS dst	Перехід, якщо результат негативний (*SF* = 1)	*IP* ← *IP* + *dst*	-//-
CALL NEAR dst	Внутрішньосегментний виклик процедури (у сегменті 64 Кбайт)	*IP* ← *dst*	-//-
CALL FAR dst	Міжсегментний виклик процедури (у межах пам'яті 1 Мбайт)	*IP* ← *dst*; *CS* ← [*dst* + 2]	-//-
RET (*NEAR*) (*n*)	Повернення з внутрішньосегментної процедури. Параметр *n* задає корекцію покажчика стеку	*IP* ← [*SS*:*SP*]; *SP* ←*SP* + 2(*n*)	-//-
RET (*FAR*) (*n*)	Повернення з міжсегментної процедури. Параметр *n* задає корекцію покажчика стеку	*IP* ← [*SS*:*SP*]; *SP* ←*SP* + 2(*n*); *CS* ← [*SS*:*SP*]	-//-

Продовження табл. 2

Мнемокод команди	Опис команди	Алгоритм команди	Зміна прапора
КОМАНДИ ПЕРЕРИВАНЬ			
INT n	Виконання програмного переривання	$SP \leftarrow SP - 2$; $[SS{:}SP] \leftarrow FLAGS$; $SP \leftarrow SP - 2$; $[SS{:}SP] \leftarrow CS$; $SP \leftarrow SP - 2$; $[SS{:}SP] \leftarrow IP$	*IF*, *TF* = 0
INTO	Виконання програмного переривання 4, якщо прапорець *OF* = 1	–	-//-
IRET	Повернення з переривання	$IP \leftarrow [SS{:}SP]$; $SP \leftarrow SP + 2$; $CS \leftarrow [SS{:}SP]$; $SP \leftarrow SP + 2$; $F \leftarrow [SS{:}SP]$; $SP \leftarrow SP + 2$	усі
КОМАНДИ КЕРУВАННЯ СТАНОМ МІКРОПРОЦЕСОРА			
CLC	Скидання прапорця перенесення	$CF \leftarrow 0$	$CF = 0$
CMC	Інверсія прапорця перенесення	$CF \leftarrow \overline{CF}$	*CF*
STC	Установлення прапорця перенесення	$CF \leftarrow 1$	$CF = 1$
CLD	Скидання прапорця напряму	$DF \leftarrow 0$	$DF = 0$
STD	Установлення прапорця напряму	$DF \leftarrow 1$	$DF = 1$
CLI	Заборона маскованих апаратних переривань	$IF \leftarrow 0$	$IF = 0$
STI	Дозвіл маскованих апаратних переривань	$IF \leftarrow 1$	$IF = 1$
HLT	Зупинка процесора	–	Не діє
WAIT	Очікування сигналу на лінії *TEST*	–	-//-
ESC dst	Передача коду команди *dst* арифметичному співпроцесору	–	-//-

Продовження табл. 2

Мнемокод команди	Опис команди	Алгоритм команди	Зміна прапора
LOCK	Префікс блокування шини на час виконання наступної інструкції у максимальному режимі	–	-//-
NOP	Немає операцій	–	-//-

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. *Абель П.* Язык ассемблера для IBM PC и программирования: Пер. с англ. – М.: Высш. шк., 1992. – 447 с.

2. *Варлинский Н.Н., Попов Е.А., Хвощ С.Т.* Микропроцессоры и микроЭВМ в системах автоматического управления: Справочник / Под общ. ред. *С.Т. Хвоща*. – Л.: Машиностроение. Ленингр. отд-ние, 1987. – 640 с.

3. *Григорьев В.Л.* Программирование однокристальных микропроцессоров. – М.: Энергоатомиздат, 1987. – 288 с.

4. *Дао Л.* Программирование микропроцессора I8088. – М.: Мир, 1988. – 354 с.

5. *Мячев А.А.* Мини- и микроЭВМ систем обработки информации: Справочник. М.: Энергоатомиздат, 1991. – 304 с.

6. *Самофалов К.Г., Викторов О.В.* Микропроцессоры. – К.: Техника, 1989. – 312 с.

7. *Фергусон Дж., Макари Л., Уилльямз П.* Обслуживание микропроцессорных систем: Пер. с англ. – М.: Мир, 1989. – 336 с.

8. *Холленд Р.* Микропроцессоры и операционные системы: Краткое справочное пособие: Пер. с англ. – М.: Энергоатомиздат, 1991. – 192 с.

Навчальне видання

ПАВЛОВ Геннадій Вікторович
ПОКРОВСЬКИЙ Михайло Володимирович

Методичні вказівки до виконання лабораторних робіт із дисципліни

"МІКРОПРОЦЕСОРНА ТЕХНІКА"

(*українською мовою*)

Редактор *Т.Б. Забабуріна*
Комп'ютерна правка та верстка *В.Г. Мазанко*
Коректор *М.О. Паненко*

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції
ДК № 2506 від 25.05.2006 р.

Підписано до друку 14.06.07. Папір офсетний. Формат 60×84/16.
Друк офсетний. Гарнітура "Таймс". Ум. друк. арк. 2,0. Обл.-вид. арк. 2,2.
Тираж 100 прим. Вид. № 4. Зам. № 92. Ціна договірна

Видавець і виготівник Національний університет кораблебудування,
54002, м. Миколаїв, вул. Скороходова, 5

Для нотаток

Для нотаток