Лабораторна робота № 7

Тема: Керування ходом виконання програми на асемблері.

Мета: Навчитися використовувати типові керуючі структури у програмах на асемблері.

Короткі теоретичні відомості

Програми рідко являють собою лінійну послідовність команд. Більшість програм, написаних на різних мовах програмування, містять точки, в яких приймається рішення про передачу керування на інші частини програми. Наприклад, структура

```
if x > 5 then y = 2 else y = 3
```

реалізується блок-схемою, представленою на рис. 4.1.:

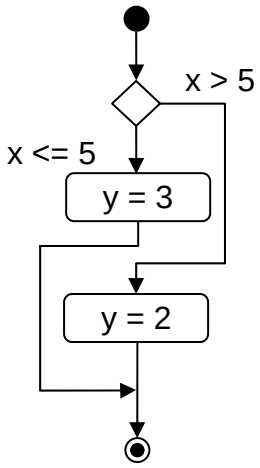

Рис. 4.1. Блок-схема керуючої структури.

1. Команди передачі керування

Є два типи команд передачі керування: безумовні та умовні. *Безумовний перехід* завжди викликає зміну послідовності виконання команд (аналог **goto**). *Умовний перехід* виконується на основі аналізу певних даних або умов (зокрема прапорців).

а) Безумовні переходи

Команда безумовного переходу перериває лінійну послідовність команд. В лічильник команд завантажується нова адреса і починає виконуватись інша ділянка програми. Команда безумовного переходу має такий синтаксис:

```
JMP [тип] операнд     ; операнд – адреса переходу або мітка
```

Операндом команди JMP найчастіше виступає мітка, на яку передається керування. Процесор має кілька машинних команд безумовного переходу. Відрізняються вони розміром адреси переходу (один байт, два байти або чотири байти). На етапі трансляції

компілятор замінює символічне ім'я на конкретну адресу переходу. Відповідно до розміру адреси переходи бувають: короткий (SHORT), близький (NEAR), далекий (FAR). Тип переходу задається модифікатором `[тип]`. Тип мітки повинен відповідати типу переходу.

Короткий перехід (двобайтна команда) може здійснювати переходи на відстань від -127 до +127 байтів від адреси поточної команди. Операндом є однобайтна адреса із знаком, яка додається до лічильника команд. При переходах назад цей тип може бути опізнаний автоматично, але при переходах вперед необхідно вказувати тип переходу явно:

```
jmp short lab_1
```

Близький перехід (трибайтна команда) здійснює переходи в межах сегменту, для чого використовується двобайтна адреса. Близькі переходи використовуються за умовчанням. Якщо тип мітки не відповідає типу переходу, то використовується модифікатор NEAR PTR.

```
jmp near ptr lab_2          ; або JMP reg16/mem16
```

Далекий перехід (п'ятибайтна команда) призначений для міжсегментних переходів. Операнд чотирибайтний. Містить адреси сегменту і зміщення. Якщо мітка всередині сегменту не описана як FAR, то використовують модифікатор FAR PTR.

```
jmp far ptr lab_3           ; або JMP reg32/mem32
```

Модифікатори адрес потрібні для того, щоб транслятор правильно визначив необхідний тип машинної команди для відповідного переходу. Можливі і такі варіанти команди JMP.

```
jmp bx
jmp near ptr [bx]
```

У першому випадку регістр містить адресу переходу, у другому регістр містить адресу комірки пам'яті, в якій знаходиться адреса переходу.

Для того, щоб явно визначити тип мітки, на яку буде здійснюватися перехід, використовують директиву LABEL. Наприклад:

```
lab_4 LABEL far
```

б) Умовні переходи

Команди умовної передачі керування приймають рішення про перехід на іншу ділянку програми на основі аналізу певних умов. Якщо умова виконується, то керування передається на адресу (мітку), визначену операндом. Якщо умова не виконується, то керування передається на наступну команду.

```
Jcc операнд      ; операнд – адреса_переходу або мітка
```

де `cc` позначає умову переходу.

Умовні переходи лише короткі (–128…+127), а починаючи з 80386 в межах сегменту коду. Умовами для прийняття рішення про перехід можуть бути:

- стан прапорців;
- порівняння операндів командою CMP;
- стан регістра CX.

Стан прапорців може бути змінений будь-якою арифметичною або логічною командою, яка впливає на їх стан. Наступні команди аналізують стан окремих прапорців і здійснюють перехід, якщо прапорець встановлений (1) або скинутий (0).

Прапорець	Стан прапорця	
	0	1
ZF	JNZ	JZ
OF	JNO	JO
SF	JNS	JS
CF	JNC	JC
PF	JNP	JP

Для того, щоб встановити прапорці часто використовують команду TEST.

Порівняння операндів здійснюється командою **CMP A, B**. Ця команда виконується аналогічно команді SUB, але результат виконання ніде не зберігається, а лише встановлюються прапорці. Команди умовних переходів, які використовують після цієї команди, аналізують стан деяких прапорців і встановлюють співвідношення між операндами. Є дві групи команд, які призначені для знакових і беззнакових операндів.

Умова переходу	Знакові операнди	Беззнакові операнди
A = B	JE	JE
A ≠ B	JNE	JNE
A < B	JL / JNGE	JB / JNAE
A > B	JG / JNLE	JA / JNBE
A ≤ B	JLE / JNG	JBE / JNA
A ≥ B	JGE / JNL	JAE / JNB

В умовних позначеннях команд умовних переходів літери означають наступне: E – equal (дорівнює), L – less (менше), G – greater (більше), B – below (нижче), A – above (вище), N – not (не). Наприклад, команда JNLE означає Jump if Not Less or Equal (перейти, якщо не «менше або дорівнює», тобто більше.

Стан регістра C аналізується командою умовного переходу **JCXZ**. Ця команда здійснює перехід на мітку, задану операндом, якщо регістр CX містить 0.

2. Команди керування циклами

Процесор 80x86 має кілька спеціальних команд для організації циклів.

Команда LOOP. Команда декрементує (зменшує на 1) регістр CX, перевіряє вміст регістра CX, якщо CX ≠ 0, то передає керування на мітку. Якщо CX=0, то виконує наступну команду. Команда LOOP не змінює прапорців.

```
cx := cx - 1
if cx ≠ 0, goto target
```

Команда LOOPE/LOOPZ. Команда діє аналогічно попередній, але додатково перевіряє стан прапорця ZF. Якщо прапорець встановлений (ZF=1), цикл продовжується. Цю команду найчастіше використовують після команди CMP. Алгоритм роботи команди такий:

```
cx := cx - 1
if ZF = 1 and cx ≠ 0, goto target
```

Команда LOOPNE/LOOPNZ. Дія команди схожа на попередню, але цикл продовжується, якщо прапорець ZF=0. Команда зручна для пошуку заданого елемента в масиві. Алгоритм роботи команди такий:

```
cx := cx - 1
if ZF = 0 and cx ≠ 0, goto target
```

3. Реалізація стандартних керуючих структур

Система команд процесорів з архітектурою IA-32 не містить умовних логічних структур, аналогічних мовам високого рівня. Проте за допомогою команд порівняння та умовних переходів можна реалізувати різноманітні логічні структури. Практично спочатку обчислюється значення умовного виразу за допомогою команд CMP (SUB) або AND (TEST), які встановлюють відповідні прапорці. Потім однією з команд умовного переходу аналізується стан певних прапорців і виконується перехід за вказаною адресою.

Наступні приклади показують, що типові структури керування досить просто реалізуються у програмах на асемблері.

C-структура	Асемблерна структура

if (a > 5) b = 2;
  else b = 3;
	cmp ax, 5
	jg thenb
	mov bx, 3 ; блок else
	jmp goon
	thenb: mov bx, 2
	goon:

sum = 0;
for (i = 10; i > 0; i--)
  sum += i;
	mov ax, 0
	mov cx, 10
	lp1: add ax, cx ; тіло циклу
	loop lp1

while (c > 0) {
  /*body of loop*/
}
	while: cmp cx, 0
	jbe endw ; if ≤
	; тіло циклу
	jmp while
	endw:

do { /*body of loop*/ } **while** (c > 0);	do: ; тіло циклу cmp cx, 0 ja do

Порядок виконання роботи

Задано текстовий рядок: "The quick brown Fox jumps over a lazy Dog". Розробити програму для обробки рядка згідно з завданням варіанту, використовуючи при цьому типові керуючі структури на асемблері. Для цього:

1. Розробити алгоритм розв’язання завдання.

2. Написати і відлагодити програму. У програмі передбачити виведення на екран заданого текстового рядка і кінцевого результату.

Варіанти завдань

1. Перетворити у рядку всі літери на великі.

2. Перетворити у рядку всі літери на малі.

3. Порахувати кількість слів у рядку.

4. Видалити у рядку всі пропуски між словами.

5. Вивести кожне слово речення окремим рядком.

6. Записати рядок у зворотному порядку.

7. Записати в рядку кожне слово з великої літери.

8. Ввести з клавіатури одну літеру. Вивести на екран порядковий номер першої такої літери в рядку.

9. Вивести на екран кількість літер **a**, **b**, **e**, **o**, **s** в рядку.

10. Порахувати кількість слів у рядку, які мають по 3 літери.

11. Порахувати кількість слів у рядку, які мають по 5 літер.

12. Порахувати кількість слів у рядку, написаних з великої літери.

Контрольні питання

1. Які ви знаєте засоби зміни послідовності виконання команд програми?

2. Які ви знаєте команди безумовного переходу?

3. Чим відрізняються близькі і далекі переходи?

4. Які команди називають командами умовного переходу?

5. Які умови можуть бути проаналізовані в командах умовного переходу?

6. Що означає мнемоніка команди JNO?

7. Чи впливає на якісь регістри процесора команда JCXZ? А команда LOOP?

8. Чи впливають на якісь прапорці команди JCXZ, JNE, LOOPZ?