Полтавський національний технічний університет імені Юрія Кондратюка

Лекція № 10
з навчальної дисципліни "Архітектура комп'ютерів"

Модуль 3. Призначення, класифікація та характеристики процесорів. Архітектура і система команд МП x86

Тема лекції:
Команди передачі управління МП I80X86. Умовні та безумовні переходи

План лекції

1. *Класифікація команд передачі управління МП I80X86. Команди безумовних переходів.*
2. *Команди умовних переходів.*
3. *Операції з прапорами.*
4. *Реалізація конструкції IF-THEN-ELSE.*
5. *Реалізація операторів-перемикачів.*

Факультет інформаційних та телекомунікаційних технологій і систем

Кафедра комп'ютерної інженерії
К.т.н., доцент Тиртишніков О.І.

1. Класифікація команд передачі управління МП I80X86. Команди безумовних переходів

Послідовний порядок виконання команд МП може бути змінений командами передачі управління. Рішення про передачу управління може бути:

- ***безумовним*** - управління передається не наступній по черзі команді, а іншій, місце розташування якої визначено адресою переходу;
- ***умовним*** - рішення про те, яка команда буде виконуватися наступною, приймається на основі аналізу деяких умов або даних.

Адреса переходу може знаходитися в поточному сегменті коду або в деякому іншому сегменті. У першому випадку перехід називається внутрішньо-сегментним, або близьким (***NEAR***), у другому - міжсегментним, або далеким (***FAR***). При внутрішньосегментному переході змінюється тільки вміст регістра ***eip/ip***, при міжсегментному - ***cs*** і ***eip/ip.*** При виконанні переходів конвеєр усередині МП скидається.

За принципом дії команди МП, що забезпечують організацію переходів у програмі, можна розділити на три групи:

1. ***Команди безумовної передачі управління:***

- команди безумовного переходу;
- виклику процедури і повернення з процедури;
- виклику програмних переривань і повернення з програмних переривань.

1. Класифікація команд передачі управління МП I80X86. Команди безумовних переходів

2. Команди умовної передачі управління:

- команди переходу по результату виконання команди порівняння ***cmp***;
- команди переходу по стану визначеного прапора;
- команди переходу по вмісту регістра ***ecx/cx***.

3. Команди управління циклом:

- команда організації циклу з лічильником ***ecx/cx***;
- команда організації циклу з лічильником ***ecx/cx*** з можливістю дострокового виходу із циклу по додатковій умові.

1.1. Команда безумовного переходу *jmp*

Синтаксис команди безумовного переходу: ***jmp [модифікатор] адреса_переходу*** ***- безумовний перехід без збереження інформації про точку повернення.***

Адреса переходу - мітка або адреса області пам'яті, у якій знаходиться покажчик переходу.

В системі команд МП є декілька кодів команд безумовного переходу ***jmp*** (три формати для типу ***NEAR*** і два формати для типу ***FAR***). Їх відмінності визначаються дальністю переходу і способом завдання цільової адреси. Модифікатор може приймати різні значення, що визначають спосіб переходу на мітку усередині поточного сегмента, або за його межами.

Наприклад, ***JMP SHORT COUNT*** – короткій перехід до мітки ***COUNT***.

1. Класифікація команд передачі управління МП I80X86. Команди безумовних переходів

Команда ***jmp*** може використовуватися для обходу якої-небудь частини програми, або для переходу до іншого сегмента.

1.2. Команди виклику процедури **call** *та повернення з процедури* **ret**

Команда ***call*** має таки ж самі формати, як команда ***jmp*** (крім команди короткого переходу), та виконується аналогічно, але адреса повернення (наступної команди) запам'ятовується у стеці (при внутришньосегментних переходах – ***ip***, при міжсегментних – спочатку ***ip***, потім – ***cs***).

Команда ***ret*** повертає управління програмі, що здійснила виклик. Кожна процедура має хоча б одну команду ***ret***. Передача управління здійснюється шляхом добування із стека адреси повернення, яка була записана відповідною командою ***call,*** тому команда ***ret*** не містить адресної інформації та неявно адресує вершину стека.

Тип команди ***ret*** має співпадати з типом команди ***call,*** яка викликала процедуру.

2. Команди умовних переходів

МП I8086 має 18 команд умовних переходів. Ці команди дозволяють перевіряти:

- відношення між операндами зі знаком ("більше - менше");
- відношення між операндами без знака ("вище - нижче");
- стан деяких арифметичних прапорів (***zf*** - прапор нуля, ***sf*** - прапор знака, ***cf*** - прапор переносу, ***of*** - прапор переповнення, ***pf*** - прапор парності) або комбінацій зазначених прапорів.

Якщо умова виконується, здійснюється перехід; якщо ні – виконується наступна у черзі команда. Відображення станів прапорів МП подано таблицею.

Прапор	Назва	Встановлений	Скинутий
CF	**Прапор переносу**	1	0
PF	**Прапор парності (паритету)**	1	0
ZF	**Прапор нуля**	1	0
SF	**Прапор знака**	1	0
OF	**Прапор переповнення**	1	0

Всі команди мають єдиний двобайтовий формат, який дозволяє здійснювати короткі переходи відносно вмісту ***ip***. У ранніх моделях МП I80X86 (до I80286 включно) команди умовних переходів здійснювали тільки переходи на відстань від -128 до +127 байт від вмісту ***ip,*** у наступних моделях це обмеження усунено, але переходи можливі тільки в межах поточного сегмента коду. ***Міжсегментні умовні переходи не допускаються.***

2. Команди умовних переходів

Команди умовних переходів мають однаковий синтаксис: ***jcc мітка_переходу***. Мнемокод усіх команд починається з "***j***" - від слова ***jump*** (стрибок, перехід), ***cc*** - визначає конкретну умову, яка аналізується командою. Значення абревіатур у назві команди ***jcc*** подані таблицею.

Мнемонічне позначення	Англ.	Укр.	Тип операндів
E e	**equal**	Дорівнює	Будь-які
N n	**not**	Ні	Будь-які
G g	**greater**	Більше	Числа зі знаком
L l	**less**	Менше	Числа зі знаком
A a	**above**	Вище, у значенні “більше”	Числа без знаку
B b	**below**	Нижче, у значенні “менше”	Числа без знаку

Для того, щоб прийняти рішення про те, куди буде передане управління командою умовного переходу, попередньо повинна бути сформована умова, на підставі якої і буде прийматися рішення про передачу управління. Джерелами такої умови можуть бути:

- будь-яка команда, що змінює стан визначених арифметичних прапорів;
- команда порівняння ***cmp***, що порівнює значення двох операндів;
- стан регістра - лічильника ***ecx/cx***.

2. Команди умовних переходів

2.1. Команда порівняння *cmp*

Команда порівняння ***cmp*** *(від compare)* так само, як і команда ***sub***, виконує віднімання операндів і встановлює відповідні прапори, але не записує результат на місце першого операнда. ***Синтаксис команди: cmp операнд_1, операнд_2 - порівнює два операнди і за результатами порівняння встановлює прапори.*** Прапори, які встановлені командою ***cmp***, можна аналізувати спеціальними командами умовного переходу. Перелік команд умовних переходів для команди ***cmp*** поданий таблицею.

Типи операндів	Мнемокод команди	Критерій умовного переходу	Значення прапорів для виконання переходів
Будь-який	**je**	операнд_1 = операнд_2	**zf = 1**
Будь-який	**jne**	операнд_1 < >операнд_2	**zf = 0**
Зі знаком	**jl /jnge**	операнд_1 < операнд_2	**sf <> of**
Зі знаком	**jle/jng**	операнд_1 <= операнд_2	**sf <> of or zf = 1**
Зі знаком	**jg/jnle**	операнд_1 > операнд_2	**sf = of and zf = 0**
Зі знаком	**jge/jnl**	операнд_1 => операнд_2	**sf = of**
Без знака	**jb/jnae**	операнд_1 < операнд_2	**cf = 1**
Без знака	**jbe/jna**	операнд_1 <= операнд_2	**cf = 1 or zf=1**
Без знака	**ja/jnbe**	операнд_1 > операнд_2	**cf = 0 and zf = 0**
Без знака	**jae/jnb**	операнд_1 => операнд_2	**cf = 0**

Видно, що однаковим значенням прапорів відповідає декілька різних мнемокодів команд умовних переходів (відділені один від одного косою рискою в таблиці).

2. Команди умовних переходів

2.2. Команди умовного переходу і прапори

Мнемонічне позначення деяких команд умовних переходів відображує назву прапора, якій вони використовують, і має наступну структуру: першим йде символ "***j***", другим - або позначення прапора, або символ заперечення "***n***", після якого розміщується назва прапора. Якщо символу "***n***" немає, то перевіряється стан прапора, і якщо він дорівнює 1, виконується перехід на мітку переходу. Якщо символ "***n***" присутній, перевіряється стан прапора на рівність 0, і у випадку вірності умови виконується перехід на мітку переходу. Мнемокоди команд, назви прапорів і умови переходів подані таблицею. Ці команди можна використовувати після будь-яких команд, що змінюють зазначені прапори.

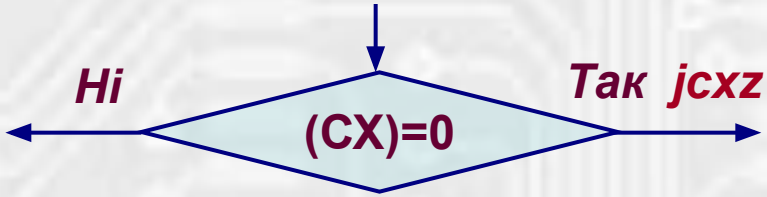
Назва прапора	Команда умовного переходу	Значення прапора для виконання переходу
прапор переносу cf	jc	cf = 1
прапор парності pf	jp	pf = 1
прапор нуля zf	jz	zf = 1
прапор знака sf	js	sf = 1
прапор переповнення of	jo	of = 1
прапор переносу cf	jnc	cf = 0
прапор парності pf	jnp	pf = 0
прапор нуля zf	jnz	zf = 0
прапор знака sf	jns	sf = 0
прапор переповнення of	jno	of = 0

2. Команди умовних переходів

2.3. Команди умовних переходів, які перевіряють вміст регістра ecx/cx

Синтаксис команди, яка перевіряє стан регістра лічильника CX: jcxz мітка_переходу (Jump if cx is Zero) - перехід, якщо cx містить нуль; jecxz мітка_переходу (Jump if ecx is Zero) - перехід, якщо ecx містить нуль.

Команда може бути корисною перед початком циклів, організованих за допомогою команди ***loop***. Так, цикл з післяумовою завжди виконується хоча б один раз, якщо навіть вміст ***CX*** дорівнює нулеві. Для запобігання "несанкціонованого" виконання циклу доцільно перед першою його командою виконати команду ***jcxz***, де в якості мітки вказана адреса комірки пам'яті, що містить команду завершення процедури ***RET.***



3. Операції з прапорами

Команди, що безпосередньо встановлюють або скидають прапорці, є безопераднними та доступні: команди скидання та встановлення – для прапорців CF, DF, IF; інвертування – тільки для CF.

Команда	Виконувана дія	Значення прапора після виконання команди
CLC	Скидання прапора: CF	CF = 0
CLD	DF	DF = 0
CLI	IF	IF = 0
CMC	Інвертування прапора переносу	якщо CF = 1, то CF = 0 якщо CF = 0, то CF = 1
STC	Встановлення прапора: CF	CF = 1
STD	DF	DF = 1
STI	IF	IF = 1

4. Реалізація конструкції IF-THEN-ELSE

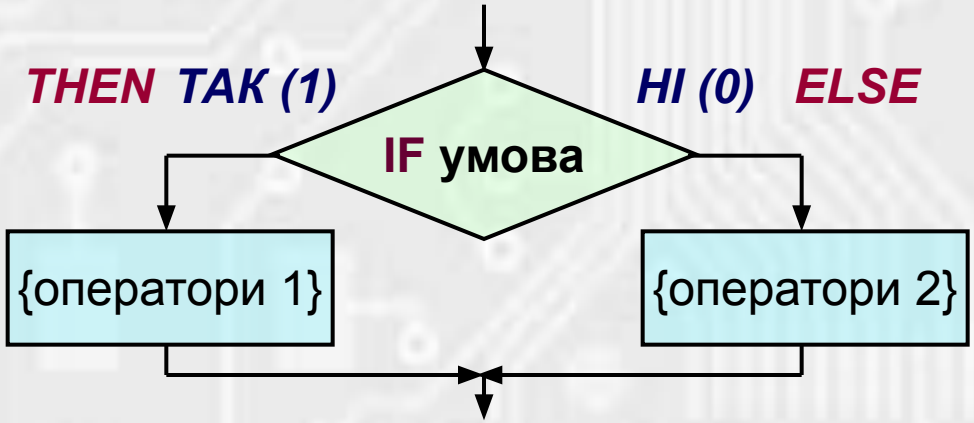


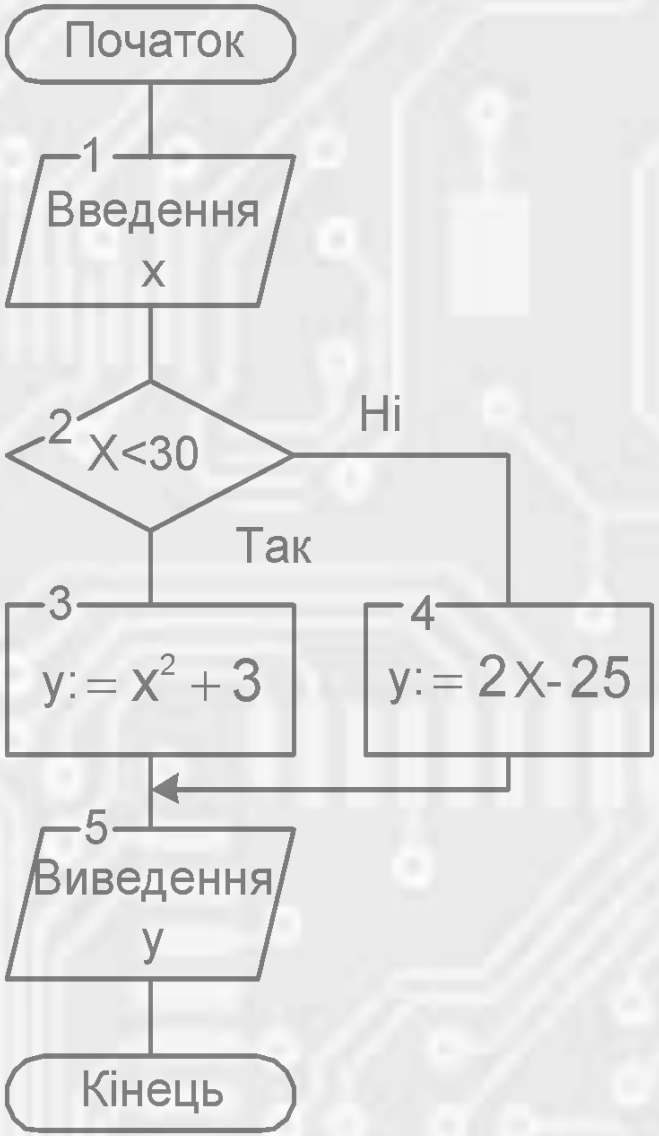
Схема реалізації умовного оператора мови ***C*** (або аналогічного оператора ***Pascal***): ***IF*** **(умова) THEN {оператори 1} ELSE {оператори 2}** на асемблері:

(Команди перевірки умови)
Jcc ELSE
(Команди, що відповідають Операторам 1)
Jmp ENDIF
ELSE: **(Команди, що відповідають Операторам 2)**
ENDIF: ***nop*** **; порожній код**

Мітки в програмі мають бути унікальними. Якщо в програмі є декілька аналогічних фрагментів, назви міток повинні відрізнятися (наприклад, ***ELSE1*** та ***ENDIF1, ELSE2*** та ***ENDIF2*** і.т.д.

4. Реалізація конструкції IF-THEN-ELSE

Приклад розгалуженої програми




```
#make_COM#
include 'emu8086.inc'
ORG 100h
CALL scan_num
      mov dl,30
      mov al,cl
      cmp al,dl
      jge else
      mul al
      add ax,3
      jmp endif
else: mov bl,2
      mul bl
      sub ax,25
endif: nop
CALL print_num
hlt
DEFINE_SCAN_NUM
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS
END
```

Функція ***SCAN_NUM*** – введення числа з клавіатури з записом в CX.

5. Реалізація операторів-перемикачів

Оператор – перемикач у мові С має вигляд:

Switch **(Керуюча змінна)** ***{***
Case **Конст1: Оператори_1**
Case **Конст2: Оператори_2**
…
default: **Оператори_n*}***

Оператор – перемикач у мові Pascal має вигляд:

case **Вираз** ***of***
Конст1: Оператор_1
Конст2: Оператор_2
…
else **Оператор_n**
end

У випадку, якщо керуюча змінна дорівнює якої-небудь з констант, виконуються всі оператори від відповідної мітки до завершення перемикача. Серед операторів може бути оператор ***break***, тоді відбувається вихід з перемикача. Якщо значення керуючої змінної не співпадає з жодною з констант, виконуються оператори блоку ***default.***

У випадку, якщо значення виразу дорівнює якої-небудь з констант, виконується оператор, записаний після відповідної мітки та відбувається вихід з перемикача. Якщо значення виразу не співпадає з жодною з констант, виконуються оператор, записаний після ключового слова ***else.*** Оператори 1…n можуть бути блочними.

Блок ***default*** або ключове слово ***else*** не є обов'язковими. У разі їх відсутності управління передається на оператор, наступний за перемикачем.

5. Реалізація операторів-перемикачів

Приклад: реалізуємо перемикач, який у мові С має вигляд:

```
Switch (a) {
Case 0: Оператори_0
Case 1: Оператори_1
Case 2: Оператори_2
default: Оператори_n}
```

Для реалізації перемикача введемо масив ***Table*** з трьох слів та ініціюємо його мітками переходів на відповідні гілки ***L0, L1, L2***. Значення міток будуть знаходитися у трьох суміжних словах. Зсув будь-якого з цих слів можна отримати додаванням зсуву ***Table*** з подвоєним значенням змінної a (зсув двох сусідніх слів відрізняється на 2).

Варіант 1 програми мовою Assembler

```
      cmp a,2          ; якщо a>2
      ja DFLT          ; перехід на DFLT
      mov BX,a
          shl BX,1          ; B=2*a
      jmp CS:Table[BX] ; перехід на мітку L
Table    dw L0, L1, L2        ; визначення
масиву
L0:  Оператори_0
L1:  Оператори_1
L2:  Оператори_2
DFLT:    Оператори_n
```

Увага! У наведеному прикладі дані визначаються безпосередньо у сегменті команд, тому необхідно адресувати їх з явним префіксом ***CS.***

5. Реалізація операторів-перемикачів

Для реалізації перемикача у ***Pascal*** – варіанті необхідно використовувати додаткові оператори ***Jmp DFLT***.

Приклад: реалізуємо перемикач, який у мові Pascal має вигляд:

case Вираз of
Конст1: Оператор_1
Конст2: Оператор_2
…
else Оператор_n
end

```
    cmp a,2          ; якщо a>2
    ja DFLT          ; перехід на DFLT
    mov BX,a
        shl BX,1         ; B=2*a
    jmp CS:Table[BX] ; перехід на мітку L
Table   dw L0, L1, L2        ; визначення
масиву
L0:  Оператори_0
        Jmp DFLT
L1:  Оператори_1
        Jmp DFLT
L2:  Оператори_2
DFLT:   Оператори_n
```

Варіант 2 програми мовою Assembler

Рекомендована література

1. *Юров В.И. Assembler. Учебник для вузов. 2-е изд. – СПб.: Питер, 2003.*
2. *Зубков С.В. Assembler для DOS, Windows и Unix. – М.: ДМК Пресс, 2000.*
3. *Митницкий В.Я. Архитектура IBM PC и язык Ассемблера: Учеб. Пособие. – М: МФТИ, 2000.*
4. *Схемотехніка електронних систем: У 3 кн. Кн. 3. Мікропроцесори та мікроконтролери: Підручник / В.І. Бойко, А. М. Гуржий, В.Я. Жуйков та ін. – К.: Вища шк., 2004.*
5. *Микропроцессорный комплект К1810: Структура, программирование, применение: Справочная книга / Ю.М. Казаринов, В.Н. Номоконов, Г.С. Подклетнов, Ф.В. Филиппов; Под ред. Ю.М. Казаринова. – М.: Высш. шк., 1990.*