

Лекция 11. Команды передачи управления. Реализация разветвлений

Команды передачи управления позволяют **принудительно изменить адрес памяти следующей команды в командном цикле процессора**. То есть, значения регистров **CS и IP**.

Это дает возможность реализовать разветвления и циклы в программной логике

В группу команд передачи управления входят:

- безусловные (JMP) и условные переходы (J...)
 - вызов и возврат из процедур (CALL, RET)
 - программные прерывания (INT)
-

Безусловные переходы: JMP

Безусловные переходы различаются:

- по типу:
 - «внутрисегментный» переход в текущем кодовом сегменте.
Изменяет **только регистр IP**
 - «межсегментный» переход в другой кодовый сегмент.
Изменяет **CS и IP**
- по способу указания адреса перехода:
 - «прямой» - адрес перехода указан в команде
 - «косвенный» - адрес перехода задан в регистре/памяти

Прямой внутрисегментный JMP

`jmp m1` ; переход на команду с симв. адресом m1
next: команда
.
.
.
.
.
m1: mov ax, bx

Машинный код внутрисегментного JMP содержит не адрес перехода, а «расстояние» перехода в байтах. Расстояние перехода в сегменте (Δ) вычисляется транслятором так:

$\Delta = \text{адрес перехода} - \text{адрес следующей за JMP команды}$

Для нашего примера это будет разница адресов: m1 – next

Выполнение процессором прямого внутрисегментного JMP:

$IP \leftarrow \text{текущий } IP + \Delta$

«Короткий» прямой внутрисегментный JMP

Указание **short** для транслятора позволяет сократить создаваемый им машинный код команды, если ваше расстояние перехода - в диапазоне от +127 до -128 байт. Транслятор будет выделять не 2 байта на расстояние, а 1 байт.

Примеры: протоколы трансляции обычного и короткого внутрисегментных JMP.

jmp n3

jmp **short** n3

```
1 0000      cseg segment use16
2          assume cs:cseg
3 0000 B8 0003    n1: mov ax,3
4 0003 EB 04 90    jmp n3
5 0006 B8 0004    n2: mov ax,4
6 0009 D1 E0      n3: shl ax,1
7 000B          cseg ends
8          end n1
```

```
1 0000      cseg segment use16
2          assume cs:cseg
3 0000 B8 0003    n1: mov ax,3
4 0003 EB 03      jmp short n3
5 0005 B8 0004    n2: mov ax,4
6 0008 D1 E0      n3: shl ax,1
7 000A          cseg ends
8          end n1
```

Косвенный внутрисегментный JMP

Новое значение для регистра IP задается для процессора **косвенно**: в регистре или в памяти.

При исполнении косвенного внутрисегментного JMP процессор заносит в IP новое значение.

- | <u>Примеры:</u> | Команда | Исполнение процессором |
|-----------------|---|-------------------------|
| □ | переход по адресу, заданному в регистре BX | |
| | JMP BX | ; IP ← BX |
| □ | переход по адресу, записанному в память по адресу ds: [si]. | |
| | JMP word ptr ds:[si] | ; IP ← слово из ds:[si] |

Прямой межсегментный JMP

Используют в программах с несколькими кодовыми сегментами

JMP **far ptr** имя сегмента:
внутрисегментный адрес

Указание **far ptr** позволяет транслятору отличить межсегментный переход от внутрисегментного

Пример: два кодовых сегмента

```
code1 segment use16
assume cs:code1
n1:  mov ax,3
    . . . . .
    jmp far ptr code2:m1
code1 ends

code2 segment use16
assume cs:code2
m1:  mov ax,10
    . . . . .
code2 ends
end n1
```


Условные переходы: Jxx

- Команда условного перехода выполняется процессором только, **если соблюдается условие перехода**. Иначе, она пропускается
- Условием перехода может быть **состояние арифметического флага** или **соотношение двух величин**
- Условные переходы бывают только одного типа - **прямые внутрисегментные**

Замечание: в системе команд в описании команд условных переходов через слэш (/) записаны альтернативные мнемоники этих команд. Использовать можно любую.

Переходы по состоянию арифметических флагов

Есть условные переходы по состоянию флагов **CF**, **SF**, **ZF**, **PF** и **OF**:

JC *m1*; переход по адресу *m1*, если **CF**=1

JNC .. ; переход, если **CF**=0

JZ ... ; переход, если **ZF**=1

JNZ .. ; переход, если **ZF**=0

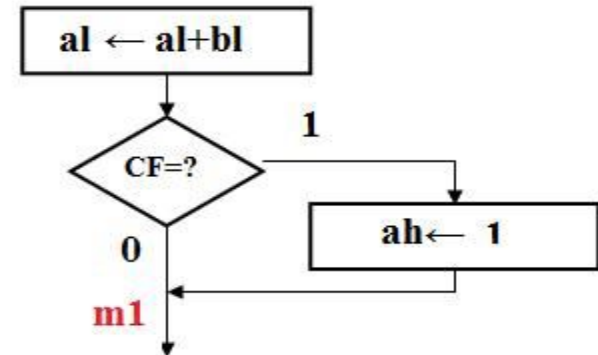
Аналогично:

JS и **JNS** – переходы по **SF**

JP и **JNP** - переходы по **PF**

JO и **JNO** – переходы по **OF**

Фрагмент алгоритма: если при сложении кодов из **AL** и **BL** возник перенос, занести в **AH** единицу.



Реализация:

```
add al, bl  
jnc ml ; переход на ml, если CF=0  
mov ah, 1  
ml: .....
```

Переходы по соотношению двух величин

- Сначала надо сравнить две величины командой CMP, что сформирует актуальные арифметические флаги
- Далее, использовать команду условного перехода по нужному соотношению величин (см. систему команд):
$$>, <, =, \neq, \Rightarrow, \Leftarrow$$

Для проверки соотношения величин процессор проверяет как отдельные флаги, так и комбинации арифметических флагов
- Команды условных переходов по соотношению различны для беззнаковых и знаковых сравниваемых величин !!

Команды условных переходов по соотношению величин

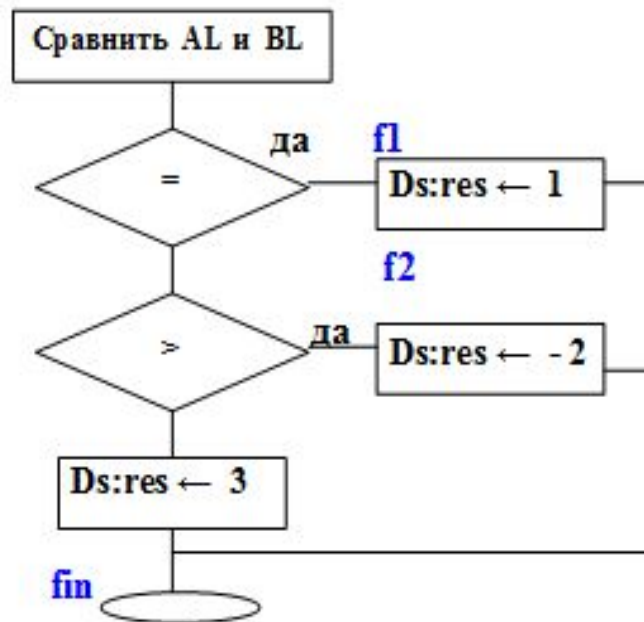
Система команд содержит две группы команд условных переходов по соотношению двух кодов: одна – для беззнаковых, другая – для знаковых.

- для соотношения беззнаковых кодов используются термины «**выше**» и «**ниже**»
- для соотношения знаковых кодов – «**больше**» и «**меньше**»
- «**равно**» - для любых кодов

Примеры:

- `cmp al, bl`
`JA m1` ; переход на m1, если $al > bl$ («**выше**»). Коды в регистрах процессор рассматривает, как беззнаковые
- `cmp al, bl`
`JG m1` ; переход на m1, если $al > bl$ («**больше**»). Коды в регистрах процессор рассматривает, как знаковые

Пример. В регистрах al и bl находятся беззнаковые коды. Если они равны, в байт памяти с адресом ds:res записать 1. Если $al > bl$, то записать число 2, иначе - 3



ВАЖНО:

Условие разветвления ставьте так, чтобы по «нет» идти в алгоритме вниз, по «да» - в бок

```
cmp al, bl
```

```
je short f1 ; если =
```

```
ja short f2 ; если >
```

```
mov ds:rez, 3
```

```
fin: mov ah, 4ch
```

```
int 21h
```

```
f1: mov ds:rez, 1
```

```
jmp short fin
```

```
f2: mov ds:rez, -2
```

```
jmp short fin
```

!! При реализации алгоритма пишите код, проходя алгоритм по «нет» до завершения. Затем – дописывайте ветки ответвлений «в бок».

Практика

Задача: Из беззнаковых кодов в регистрах al, ah и bh определить наибольший и сохранить в регистре dh.

1. Структура программы – односегментная. Кодовый сегмент cod, указатель сегмента – cs.
 2. Размещение данных: в регистрах al, ah и bh – исходные, в регистре dh – результат (наибольшее значение)
 3. Детальный алгоритм (максимально эффективный)
 4. Исходный текст
-