Тема 5. Инструкции за преход и цикъл

Инструкциите за предаване на управлението (за преход) са най-прекият начин за промяна на реда на изпълнение на инструкциите от едно място на програмата в друго. Те променят стойността на брояча на адресите (EIP/IP) с отместването на инструкцията, към която се предава управлението, а при преход и в друг сегмент – и на CS с новия сегментен адрес.
Има два основни вида инструкции за преход – за безусловен преход и за условен преход.
Инструкция за безусловен преход е инструкцията JMP:

JMP цел: предава безусловно управлението към адреса, посочен от операнда. Преходите могат да бъдат близки (в границите –32768 до +32767 B от инструкцията след прехода), къси (в границите –128 до +127 B) или далечни (в друг кодов сегмент). При къси и близки преходи операндът задава нов адрес за IP, а при далечни – нови адреси за IP и CS.
Адресът за преход е етикет или адрес в паметта, където се намира указателя за преход.

Битовете на флаговия регистър EFLAGS / FLAGS (32 / 16) имат определено функционално предназначение, отразяват особеностите на резултата от изпълнението на аритметичните или логическите команди, което прави възможно да се анализира състоянието на изчислителния процес и съответното реагиране с помощта на командите за условен преход и извикване на подпрограми.

X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

четност

CF (Carry Flag) PF (Parity Flag) AF (Auxiliary Carry Flag) ZF (Zero Flag)
SF (Sign Flag) TF (Trace Flag) IF (Interrupt Flag) DF (Direction Flag)
OF (Overflow Flag)

Осемнадесетте команди за условен прехода реализират преход при изпълнение на определено условие. Такова условие е състоянието на флаговия регистър. Флаговият регистър няма име и не може да фигурира в командите. (С изключение на командите PUSHF/ POPF, съхраняващи и възтановяващи този регистър от стека, също така и LAHF/ SAHF, копираща флаговете от регистъра в AH и обратно).
Командата CMP сравнява два операнда и въздейства на флаговете AF, CF, OF, PF, SF, ZF.
връща флаг **CMP** Сравняване (CoMPare) **CMP *операнд_ 1, операнд_ 2***
Изважда *операнд_ 2* от *операнд_ 1*. Резултатът не се запомня, но флаговете се установяват.

Флаговете, установявани с командата *cmp* могат да бъдат анализирани от специални команди за условен преход.

Команда	Стойности на флаговите регистри
JE	ZF=1
JNE	ZF=0
JL / JNGE	SF<>OF
JLE / JNG	SF<>OF или ZF=1
JG / JNLE	SF=OF и ZF=0
JGE / JNL	SF=OF
JB / JNAE	CF=1
JBE / JNA	CF=1 или ZF=1
JA / JNBE	CF=0 и ZF=0
JAE / JNB	CF=0

Преход при беззнакови данни

JE/JZ	Jump if Equal (Zero)	ZF=1
JNE/JNZ	Not Equal (Non Zero)	ZF=0
JA/JNBE	Above (not Below nor Equal)	ZF=0 и CF=0
JAE/JNB	Above or Equal (not Below)	CF=0
JB/JNAE	Below (not Above nor Equal)	CF=1
JBE/JNA	Below or Equal (Not Above)	ZF=0 или CF=1

Преход при знакови данни

JE/JZ	Jump if Equal (Zero)	ZF=1
JNE/JNZ	Not Equal (Non Zero)	ZF=0
JG/JNLE	Greater (not Less nor Equal)	ZF=0 и SF=0 и OF=0
JGE/JNL	Greater or Equal (not Less)	SF=0
JL/JNGE	Less (not Greater nor Equal)	SF=1
JLE/JNG	Less or Equal (not Greater)	ZF=0 или SF=1

Преход според специални аритметични проверки

JS	Jump if Sign	SF=1
JNS	No Sign	SF=0
JC	Cary	CF=1
JNC	No Cary	CF=0
JO	Overflow	OF=1
JNO	No Overflow	OF=0

За разлика от разгледаните вече команди за условен преход, следващата команда проверява не флаговия регистър, а стойността на регистър CX.

JCXZ <етикет> (Jump if cx is Zero) Преход, при CX=0;
JECXZ <етикет> (Jump Equal ecx Zero) Преход, при ECX=0;
Етикетът, посочен като операнд, трябва да е къс.

Вместо последователността от команди DEC CX / JNZ <етикет> се използва командата:

LOOP <етикет>
Командата позволява организация на цикли, подобни на циклите for в езиците от високо ниво с автоматично намаляване брояча на цикъла. Работата на командата се състои в изпълнението на следните действия:

- Декремент на регистър ecx/cx;
- Сравнение на регистър ecx/cx с нулата:
  - при (ecx/cx) > 0, управлението се предава на етикета за преход;
  - при (ecx/cx) = 0, управление се предава на следващата след loop команда.

Командата JCXZ обикновено се използва преди цикъла LOOP, за да предотврати изпълнението на цикъла при CX равно на нула.

Има две допълнителни команди, които освен стойността на CX проверяват и тази на ZF.
LOOPE /LOOPZ (Loop till cx <> 0 or Zero Flag = 0). Командите loope и loopz са идентични. Те работят по следния алгоритъм:

- декремент на регистър ecx/cx;
- сравняване на регистър ecx/cx с нулата;
- анализ състоянието на нулевия флаг zf:
  - ако (ecx/cx) > 0 и zf = 1, управлението се предава на етикета за преход;
  - ако (ecx/cx) = 0 или zf = 0, управлението се предава на следващата след loop команда.

LOOPNE/LOOPNZ (Loop till cx <> 0 or Not Zero flag=0). Командите loopne и loopnz са идентични. Те работят по следния алгоритъм:

- декремент на регистър ecx/cx;
- сравняване на регистър ecx/cx с нулата;
- анализ състоянието на нулевия флаг zf:
  - ако (ecx/cx) > 0 и zf = 0, управлението се предава на етикета за преход;
  - ако (ecx/cx)=0 или zf=1, управлението се предава на следващата след loop команда.

Командите LOOP / LOOPE / LOOPZ / LOOPNE / LOOPNZ, както и JCXZ в качеството на брояч могат да използват само регистър CX.

Недостатък на командите за организация на цикъл **loop, loope/loopz** и **loopne/loopnz** е, че те реализират само къси преходи (от –128 до +127 байта). За реализация на дълги цикли се налага използване на командите за условен преход и на командата *jmp*.

Тема 6. Видове адресиране

Съществуват няколко режима за адресация на данни.
Многото на брой комбинации могат да се сведат до няколко основни вида адресиране.
Ето как елементите на операнда в паметта изглеждат в общ формат:

BX	**SI**	**offset**
или +	**или +**	**Отместване**
BP	**DI**	
(база)	**(индекс)**	

Всяка от трите части на операнда в паметта не е задължителна, макар че е очевиден фактът, че сте длъжни да използвате поне един от трите елемента, защото иначе няма да получите адрес в паметта. Така се получават 16 съществуващи начина за задаване на адреса в паметта:

[offset]	**[bp+offset]**
[bx]	**[bx+offset]**
[si]	**[si+offset]**
[di]	**[di+offset]**
[bx+si]	**[bx+si+offset]**
[bx+di]	**[bx+di+offset]**
[bp+si]	**[bp+si+offset]**
[bp+di]	**[bp+di+offset]**

Тук отместването може да се сведе до 16-битова постоянна стойност.
От този списък се вижда, че всички режими на адресация се получават само от няколко елемента, комбинирани по различен начин.

debigger- td 4_3