Методи на адресация в 64-разрядните процесори на микропроцесорната фамилия Intel x86

Кирил Иванов

Април 2021 година

1. Неявна адресация (или **адресация по подразбиране**)

Адресът на операнда се подразбира от самата инструкция, т. е. номерът на инструкцията еднозначно определя адреса на операнда.

Пример (това е само инструкция):

```
sdf
```

Неявната адресация е удобна, когато се налага използването на операнди с точно фиксирани местоположения. Обикновено операндите са регистри или техни части с уникално предназначение (например флагове).

2. Адресация с автоувеличение или с автонамаление

Този вид адресация се отбелязва *понякога* като самостоятелен вид, защото по много удобен начин съчетава идеята за подразбиране на местоположението на операндите с възможността за различни обработвани операнди в различни моменти на изпълнението на една и съща инструкция.

Характерното тук е, че в съответните машинни инструкции няма никакво указание за местоположението на операндите, защото техните адреси по подразбиране се вземат от регистрите ***rsi*** (при четене) или ***rdi*** (при запис). Обаче пак по същата причина в различни моменти от изпълнението на такава инструкция е възможно да се обработват различни операнди (защото може да бъде различно съдържанието на ***rsi*** и ***rdi***).

Съответно термините **авто...** показват, че при изпълнението на такава инструкция използваните регистри ***rsi*** и ***rdi*** се променят автоматично за насочване към следващата или предхождащата (т. е. имаща съседен адрес) данна със същата разрядност като обработената. Посоката за промяната се определя от флага **DF**. При **DF=0**, се минава към следващата данна (автоувеличение), а когато **DF=1**, се минава към предхождащата данна (автонамаление).

Пример (това е само инструкция):

```
repnz movsq
```

Автоматичната промяна на адресни регистри позволява инструкциите с тази адресация да се изпълняват многократно, т. е. да работят като цикъл от една инструкция. По този начин една инструкция може да обработва цяла редица от данни с еднаква разрядност. Такъв цикъл е много пъти по-ефективен, отколкото съответния на него, организиран чрез няколко машинни инструкции. За указване на зацикляне в асемблерния език се използва префикс ***rep***, а в машинния формат на инструкцията се добавя префиксен байт. За брояч на повторенията се привлича регистърът ***rcx***. Възможно е подобни зацикляния да се прекратяват, освен при нулиране на брояча ***rcx***, още и според стойността на флага **ZF** за нулев резултат, което се указва със специфични префикси.

3. Непосредствена адресация

Операндът е част от инструкцията.

Схема на непосредствена адресация

Пример (вторият операнд):

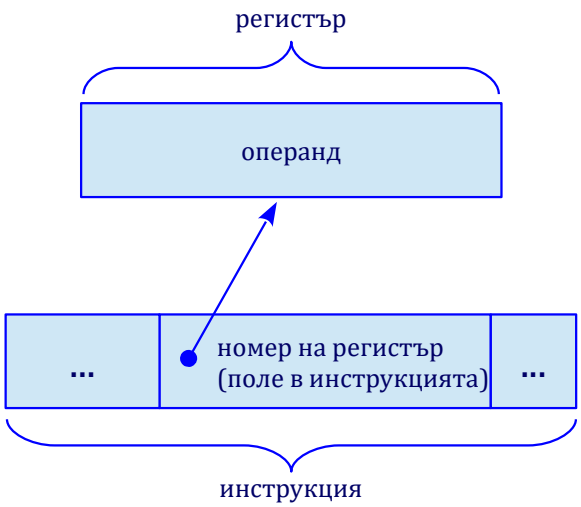
```
mov dword [Label21], -105
```

Непосредствената адресация е удобна за операнди литерали.

4. Регистрова адресация

Операнд е съдържанието на регистър.

Схема на регистрова адресация



Пример (първият операнд):

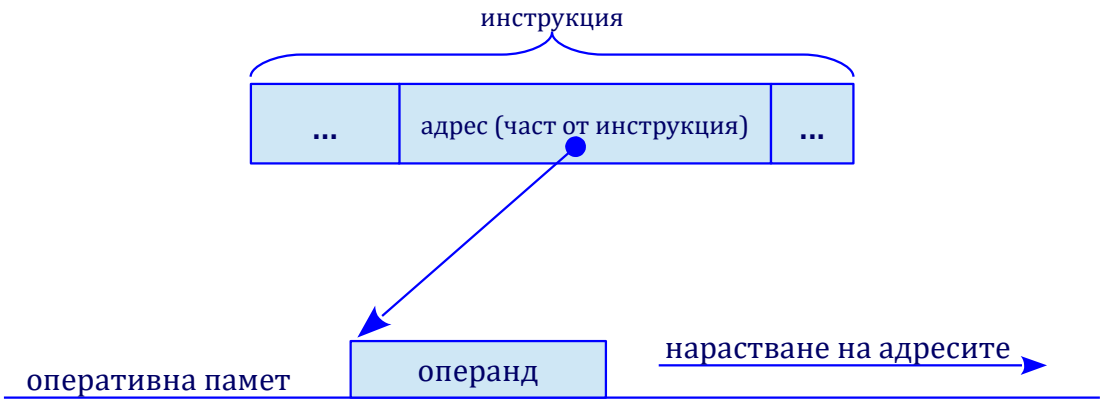
```
mov rcx, -105
```

Регистровата адресация е удобна за междинни стойности и за често използвани данни (регистровата памет е пределно бърза, по-бърза може да бъде само регистрова памет в още по-бърз процесор).

5. Пряка адресация

Адресът на операнда е пряко назован в инструкцията, той е част (поле) от инструкцията.

Схема на пряка адресация



Пример (първият операнд):

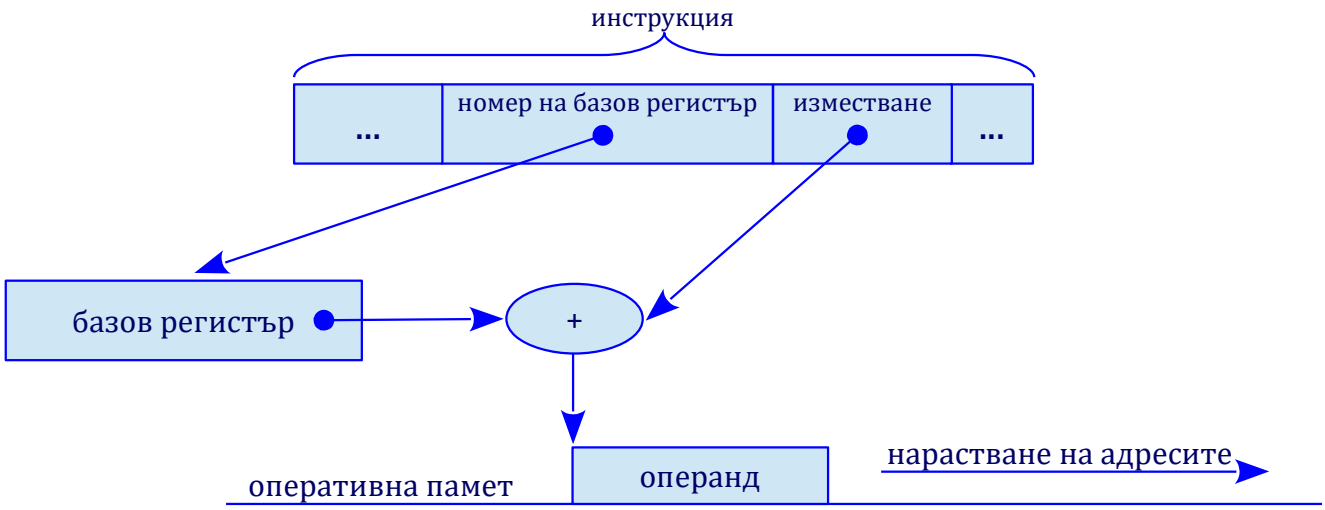
```
mov [Label23], rax
```

Пряката адресация е удобна за достъп чрез етикети или до системни константи, разполагани винаги на едни и същи адреси. Обаче инструкцията винаги работи с една и съща данна, което ограничава гъвкавостта на алгоритъма.

6. Базова с изместване адресация

Адресът на операнда е сума от съдържанието на базов регистър и изместване. В инструкцията се съдържат номерът на регистъра и изместването. Сумирането на базовия регистър и изместването става

Схема на базова с изместване адресация



по време на изпълнението на инструкцията.

Пример (първият операнд):

```
mov [rsi + 8], rax
```

Базовата с изместване адресация е удобна за достъп до полетата на структури.

7. Индексна с мащабиране и изместване адресация

Мащабиране на индексен регистър означава умножение на регистъра с число мащаб. За да се ускори този процес се използват мащаби степени на двойката. Така умножаването се свежда до поразрядно изместване наляво на толкова позиции, колкото е степенния показател.

На ниво асемблерен език адресът на операнда е сума от базов адрес, изместване и произведение на индексен регистър с мащаб. Мащабирането (умножението на индексния регистър с мащаба и сумирането на полученото произведение с базата стават *по време на изпълнението* на инструкцията.

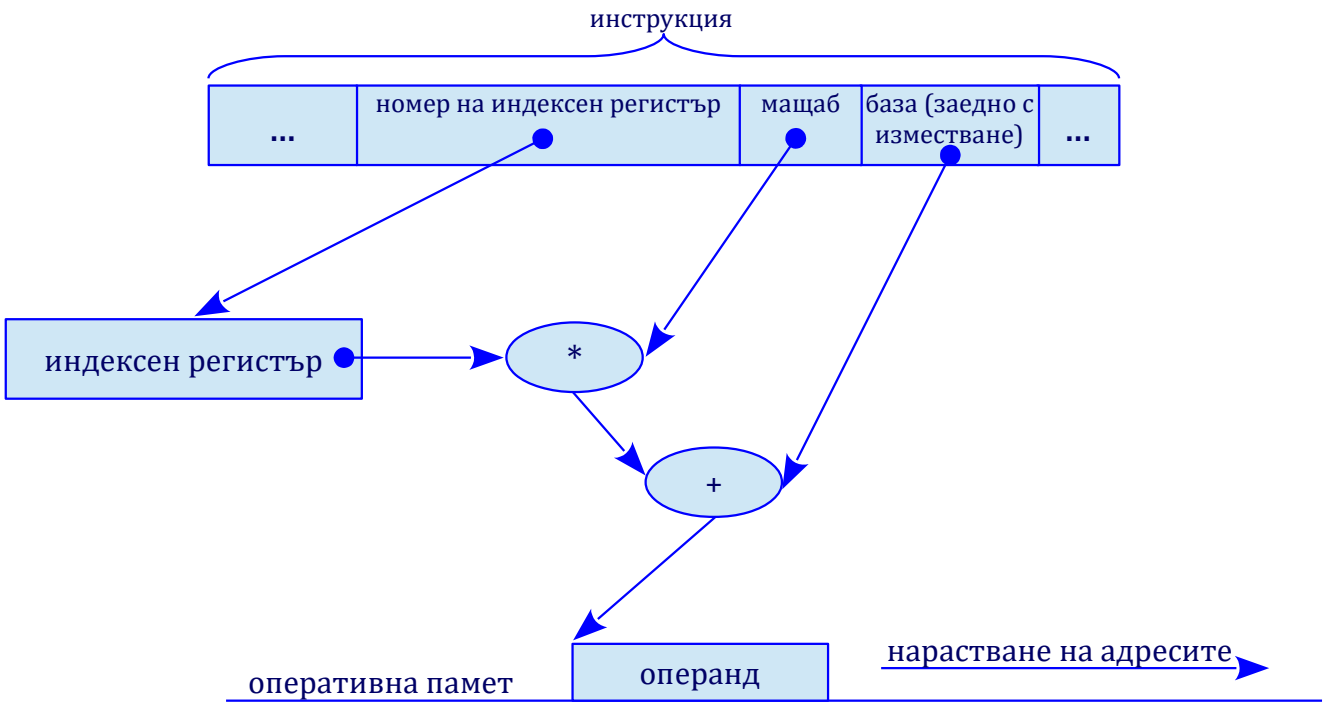
На физическо ниво адресът е сума от произведението на индексен регистър с мащаб и изместване.

Сумирането на базата и изместването, написани в асемблерния код, става *по време на транслация* и резултатът се интерпретира като изместване, записвано в инструкцията.

В машинната инструкция се съдържат (като самостоятелни полета) номерът на индексния регистър, мащабът и изместването.

Мащабирането, т. е. умножаването по мащаб, е възможно само по време на изпълнение на съответната инструкция, когато е известна стойността на индексния регистър. Разрешените мащаби са 1, 2, 4 и 8. При мащаб 1 тази адресация физически е идентична с базовата с изместване.

Схема на индексна с мащабиране и изместване адресация



Пример (първият операнд):

```
mov arrLabel5[8*rsi - 8], rax
```

Индексната с мащабиране и изместване адресация е удобна за достъп до масиви, включително и от структури. Тя е най-подходяща, когато елементите на масива заемат по 1, 2, 4 или 8 байта. Такива са най-често използваните масиви – от знакове, от числа (или или с плаваща запетая в 4 и 8 байта) и от адреси (указатели или референции).

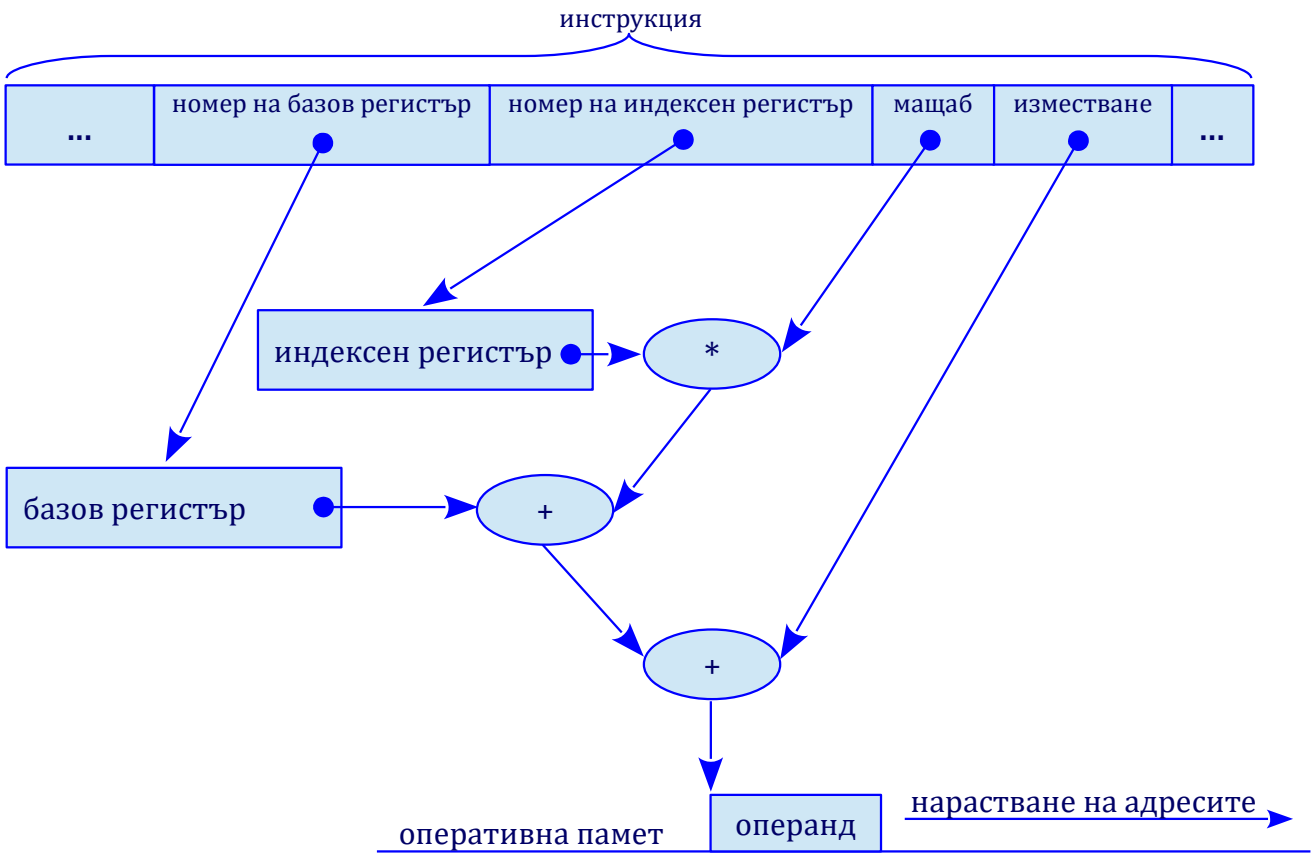
8. Базово-индексна с мащабиране и изместване адресация

Адресът на операнда е сума от базов регистър, изместване и произведение на индексен регистър с мащаб.

Изчисляването на адреса на операнда може да става само *по време на изпълнението* на инструкцията.

В инструкцията се съдържат (като самостоятелни полета) номерът на базовия регистър, изместването, номерът на индексния регистър и мащабът. Възможните мащаби са 1, 2, 4 и 8, точно както и при индексната адресация с мащабиране и изместване.

Схема на базово-индексна с мащабиране и изместване адресация



Пример (първият операнд):

```
mov [edi + 4*rsi - 12], eax
```

Базово-индексната с мащабиране и изместване адресация е удобна за достъп до двумерни масиви, включително и от структури, но пак като предишната е най-подходяща за масиви, чиито елементи заемат по 1, 2, 4 или 8 байта (каквито са адресите, знаковете и повечето числа).