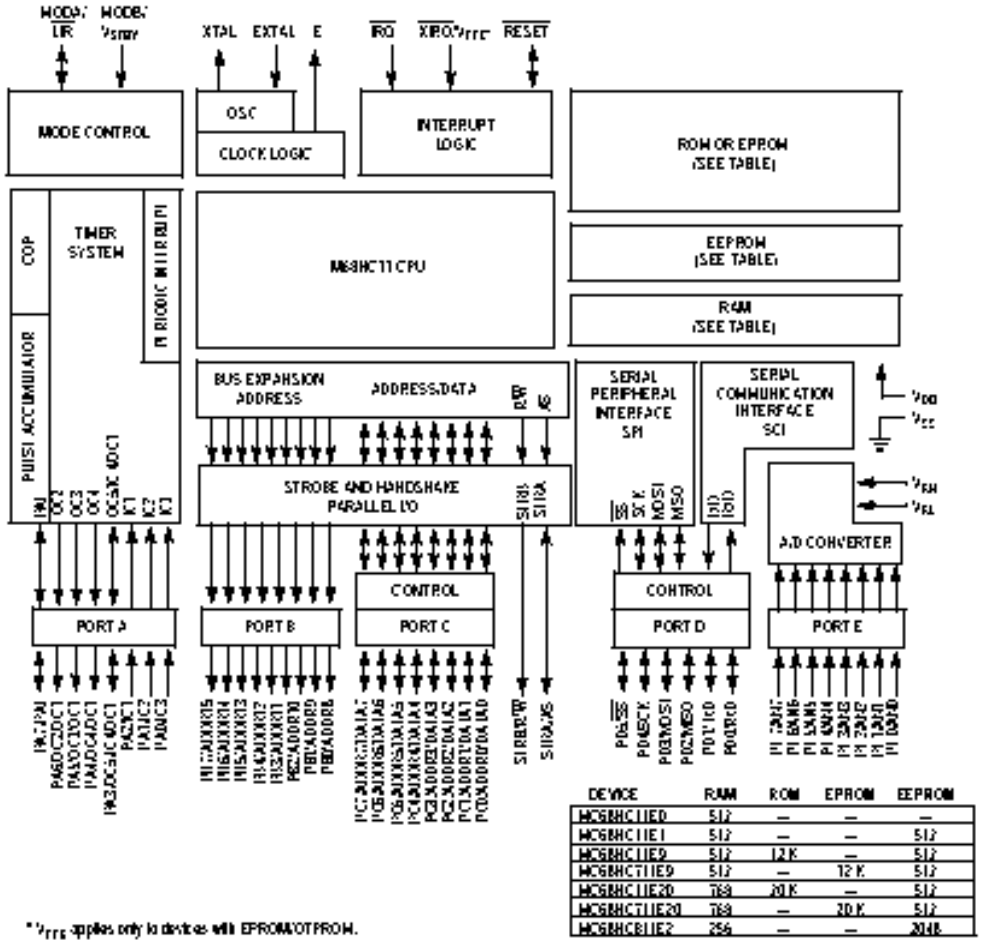
Упражнение 2

Програмен модел на MC68HC11

MC68HC11 е фамилия от едночипови микрокомпютри, чиято блокова схема е представена на фиг.1.



Фиг.1 Блокова схема на едночипови компютри MC68HC11.

MC68HC11 може да работи в следните режими:

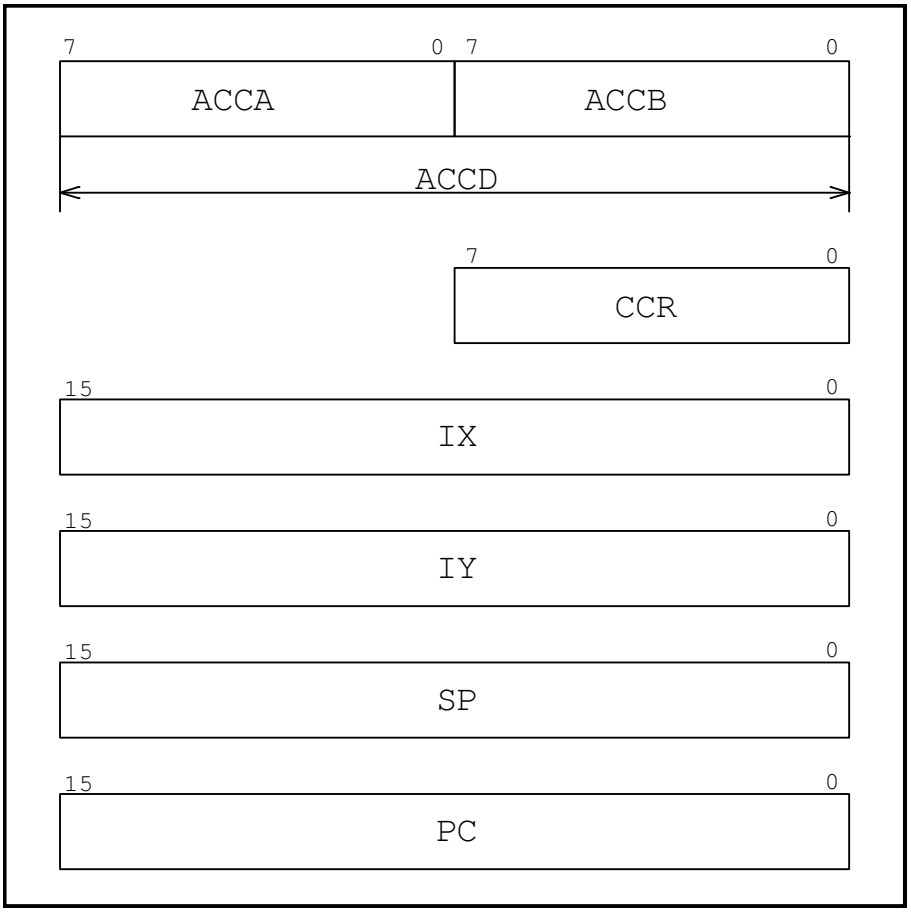
- Едночипов компютър. В този режим са достъпни само вградените в него апаратни ресурси.
- Разширен мултиплексиран. Магистралите данни, адреси и управление на процесора са достъпни на изводите на PORT B и PORT C. С цел да се намали броят на необходимите изводи, магистралите са мултиплексирани. В този режим е възможно да бъдат добавяни директно към магистралите допълнителни компоненти - RAM, ROM, интерфейсни схеми и др. Вградените в MC68HC11 ресурси също са на разположение на потребителя.

- Режим „Тест”. Дава възможност за конфигуриране на вградените в MC68HC11 ресурси - например изключване на вградените ROM, EEPROM от адресното пространство, блокиране на WATCH DOG .
- Bootstrap. Този режим е предназначен за настройка на апаратната част и програмното осигуряване.

Ядрото на MC68HC11 е осембитов процесор със следните основни характеристики:

- Разрядност на магистрала „Данни” – 8 бита ;
- Разрядност на магистрала „Адреси” – 16 бита ;
- Адресно пространство – 64K bytes;
- Разрядност на операндите – 8 бита (за някои операции - 16 бита);
- Възможност за обработка на бит;
- Вградени операции умножение и деление;
- Тактова честота на процесора – 2 MHz;

I. Регистри.



Фиг.2 Регистри на MC68HC11.

Програмно достъпните регистри на микропроцесора са представени на фиг. 2. Предназначението им е следното:

I. 1. ACCA – Акумулатор A; ACCB – Акумулатор B.

ACCA и ACCB са осембитови регистри с общо предназначение. Служат за временно съхранение на данни, с които могат да се извършват различни аритметични и логически операции. Характерна тяхна особеност е, че при операции с два операнда, акумулаторът е източник на операнд и приемник на резултата.

Например, ако трябва да бъдат сумирани M и N и резултатът е K, т.е:

N+M =K,

единият операнд, например N, трябва да са намира в акумулатора. Вторият операнд може да бъде разположен в паметта. Резултатът от действието ще се запише автоматично в същия акумулатор:

ACCA + M –> ACCA

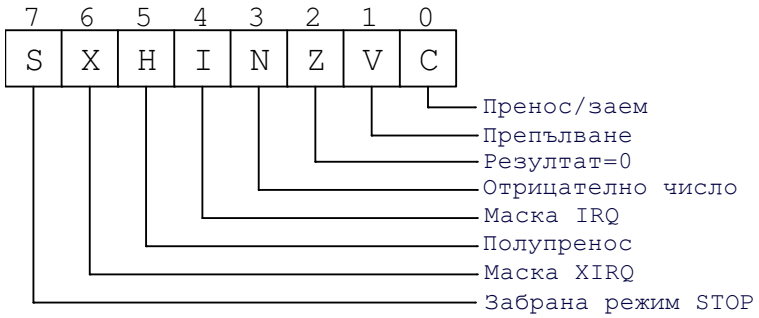
Аналогично:

ACCB + M –> ACCB

При някои операции ACCA и ACCB могат да бъдат използвани като 16-битов регистър (ACCD).

I.2. CCR – Condition Code Register.

Нарича се регистър на условията, или флагов регистър – фиг. 3.



Фиг.3 Регистър на условията.

Битове **C, V, N, H** се установяват автоматично след всяко аритметично или логическо действие и показват характера на получения резултат:

- **C – C**arry. При сумиране показва, че има пренос от най-старшия разряд на байта. При изваждане играе ролята на заем.
- **V -** O**v**erflow. Показва, че се е получило препълване на разрядната решетка. Необходим е при действия с числа със знак. Показва, че стойността на знаковия бит на резултата не е вярна.
- **Z – Z**ero. Показва, че резултатът от последното действие е 0.

- **N -** Negative. Установява се в 1,ако резултатът е отрицателно число. (Съвпада с най-старшия бит на резултата).
- **H - H**alfcarry. Полупренос. Този бит показва, че е имало пренос от младшите 4 към старшите 4 бита на байта.

Битове **I, X, S** могат да бъдат установявани принудително от програмата. Използват за управление на поведението на процесора, както следва:

- **I** – маска за прекъсване. При I=1 процесорът не възприема постъпилите заявки за прекъсване на вход IRQ.
- **X** – маска за прекъсване При X=1 процесорът не възприема постъпилите заявки за прекъсване на вход XIRQ. X може да бъде установен програмно само в 0. Установява се в 1 при Reset.
- **S -** При S=1 процесорът не изпълнява команда „STOP". („STOP" спира тактовия генератор на процесора. Нормалната работа на генератора се възстановява след Reset).

I. 3. IX – индексен регистър X; IY - индексен регистър Y.

Индексните регистри играят роля, подобна на индексната променлива **i** при достъп до елементите на масив M[i], с тази разлика, че те не показват поредния номер на елемента в масива, а съдържат адреса му.

I.4. SP – Stack Pointer. - Указател на стека.

Стекът в MC68HC11 се разполага на произволно място в RAM и се запълва от старшите към младшите адреси. SP съдържа адреса на първия свободен байт от стека. Стекът се използва за системни нужди – за съхраняване на възвратния адрес при преход към подпрограми и на статуса на прекъснатата програма при прекъсване. Потребителят също може да го използва с цел временно съхраняване на данни.

I.5. PC – Program Counter. - Програмен брояч.

Съдържа адреса на текущия байт от изпълняваната в момента програма. Увеличава се автоматично с 1 при всяко четене на байт от програмата. Командите за условен или безусловен преход, изпълнението на подпрограми и обработката на прекъсване променят стойността му.

II. Методи за адресация.

Всеки процесор може да изпълнява определена съвкупност от машинни команди, които формират неговия машинен език. Това са команди за аритметични и логически действия, за обмен на данни между паметта и процесора, команди за управление на хода на програмата и др. Програма, съставена от машинни команди се нарича машинна програма.

С редки изключения, всяка програма, независимо от това на какъв език е съставена, се превежда до машинен език и едва тогава се изпълнява от процесора.

Машинният език е специфичен за даден процесор. Машинна програма, предназначена за процесори от определен тип не може да бъде изпълнявана от процесори от друга фамилия. Например програма, предназначена за MC68HC11 е неразбираема за процесори Pentium.

Форматът на преобладаващата част от машинните команди на MC68HC11 е даден на фиг. 4.

Фиг. 4. Формат на командите.

Командата се състои от две полета:

- Първото от тях (КОП) съдържа код на операцията. За операциите, използващи IY или ACCD кодът на операция е с дължина два байта, за останалите – един байт.

- Второто поле специфицира операнда, който участва в съответната операция. То е с дължина до 2 байта и може да липсва при някои команди.

Механизмът за достъп до операнда са нарича метод на адресация.

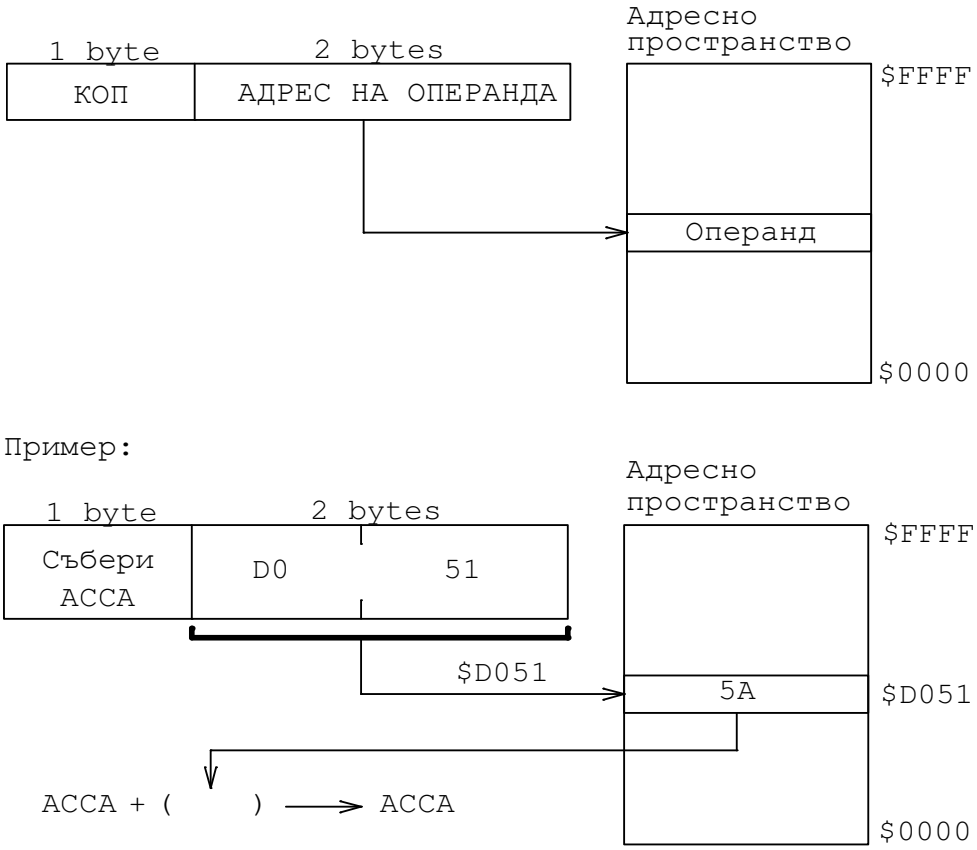
За MC68HC11 са възможни следните методи за адресация:

- Разширена;
- Директна;
- Индексна;
- Непосредствена;
- Относителна;
- Вътрешна (неявна).

Изключение от дадения на фиг.4 формат правят командите за битови операции, които използват комбинация от няколко метода за адресация - т. II.7.

II.1 Разширена адресация.

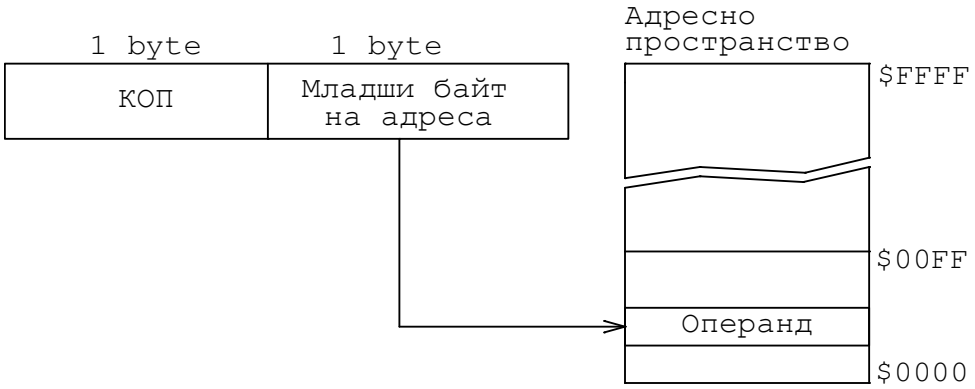
Форматът на командите е представен на фиг. 5. Полето „Операнд” на машинните команди съдържа 16-битов адрес на операнда. Процесорът го използва, за да определи разположението на операнда в адресното пространство.



Фиг. 5. Разширена адресация.

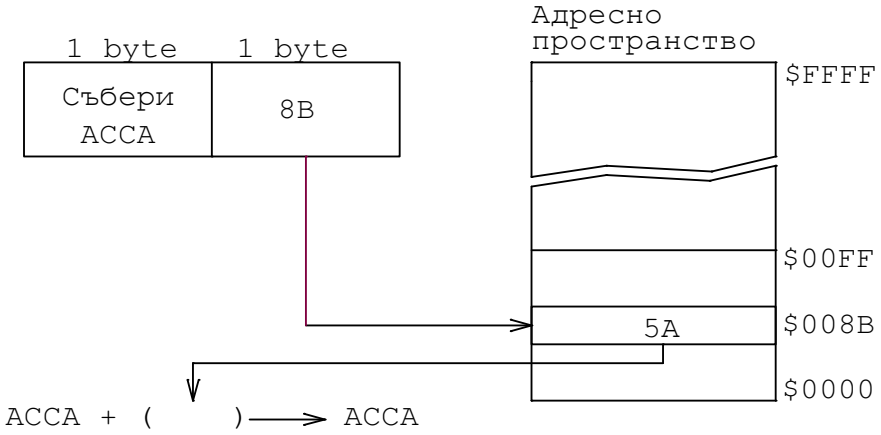
II.2. Директна адресация.

Представена е на фиг.6. Тази адресация е приложима само за операнди, разположени в първите 256 байта на адресното пространство. Нарича се още адресация в нулева страница. Полето „Операнд" е с дължина един байт и съдържа младшите осем бита на адреса на операнда. Целта на този тип адресация е да осигури по-ефективен механизъм за достъп до част от адресното пространство. Машинните команди, използващи директна адресация са по-къси и се изпълняват по-бързо, отколкото тези с разширена адресация. Програмистът ще създаде по-ефективна програма, ако разположи най-често използваните променливи в нулевата страница и използва директна адресация за достъп до тях.



Фиг. 6. Директна адресация.

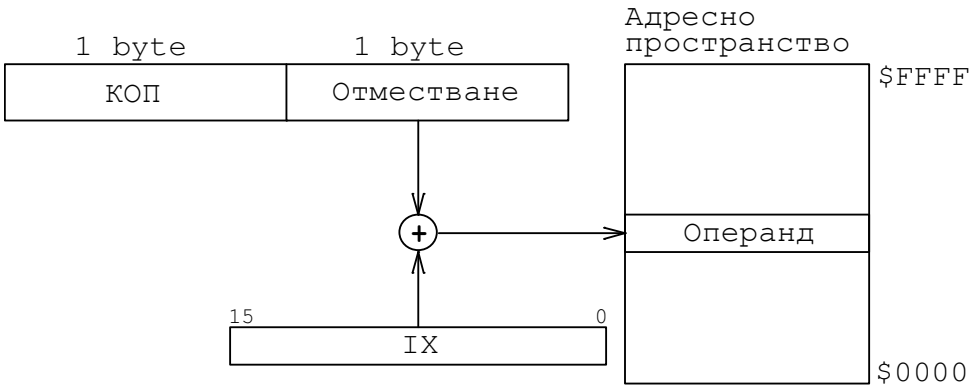
Пример:



II.3 Индексна адресация. Представена е на фиг. 7.

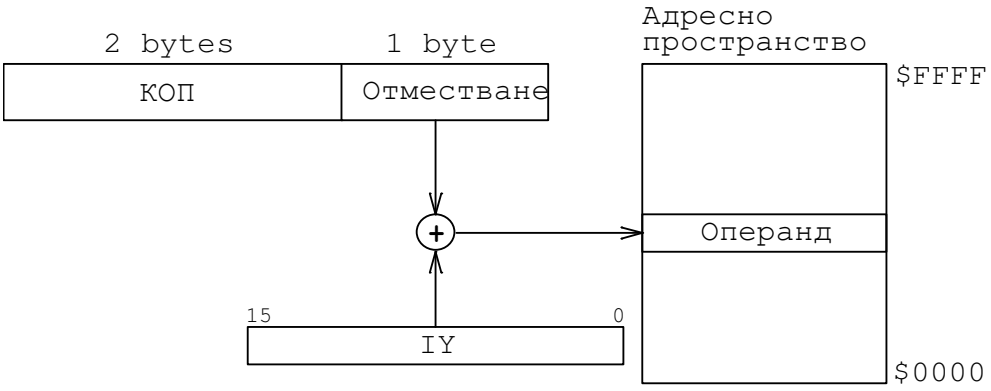
Достъпът до операнда в този случай се осъществява посредством индексния регистър IX или IY. Адресът на операнда е сумата от съответния индексен регистър и полето „Отместване" на машинната команда. Отместването е осембитово число без знак.

А) Спрямо IX



Фиг. 7А. Директна адресация спрямо IX.

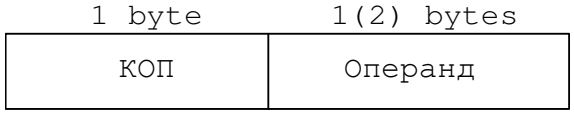
В) Спрямо IY



Фиг. 7Б. Директна адресация спрямо IY.

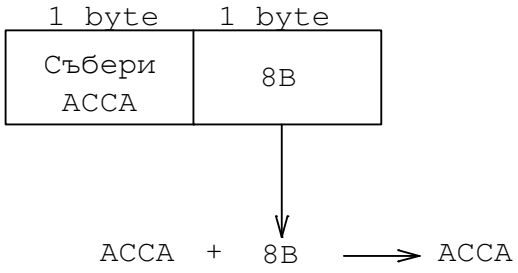
II.4 Непосредствена адресация. Представена е на фиг. 8.

Този метод на адресация дава възможност за ефективна работа с константи. Стойността на константата е записана в полето „Операнд" на машинната команда. Константата е еднобайтова, ако участва в операция с еднобайтов регистър на процесора и двубайтова – при за операция с двубайтов.



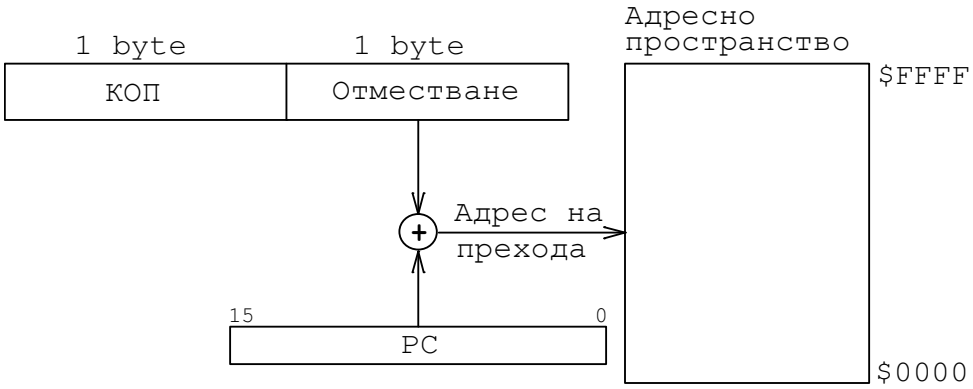
Фиг. 8. Непосредствена адресация.

Пример:



II.5 Относителна адресация – фиг. 9.

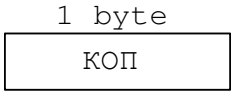
Използва се в командите за преход. Адресът на следващата команда се изчислява относно програмния брояч (PC) и е сума от съдържанието му (в момента на изпълнение на командата) и полето „Отместване" на машинната команда. Отместването е еднобайтово число със знак представено в допълнителен код. При положително отместване се извършва преход напред, а при отрицателно – връщане назад спрямо текущата команда.



Фиг. 9. Относителна адресация.

II.6.Неявна (вътрешна) адресация – фиг. 10.

В този случай машинната команда съдържа само код на операция. Той специфицира неявно операнда, с който ще се извърши предписаното действие. Такива са например командите: „ Нулирай АССА", „Инвертирай АССВ", "Увеличи с единица IX" и др.



Фиг. 10. Неявна (вътрешна) адресация

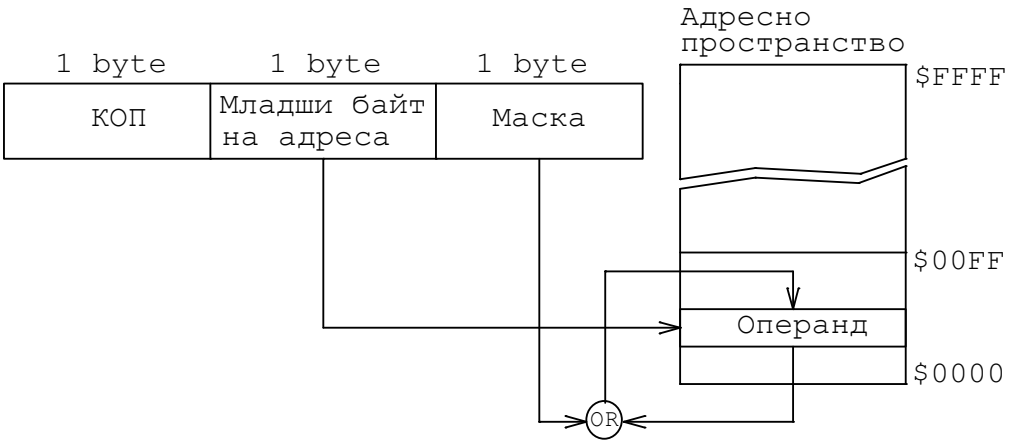
II. 7. Битови операции.

II.7.А Операции за установяване на бит в „1" или „0".

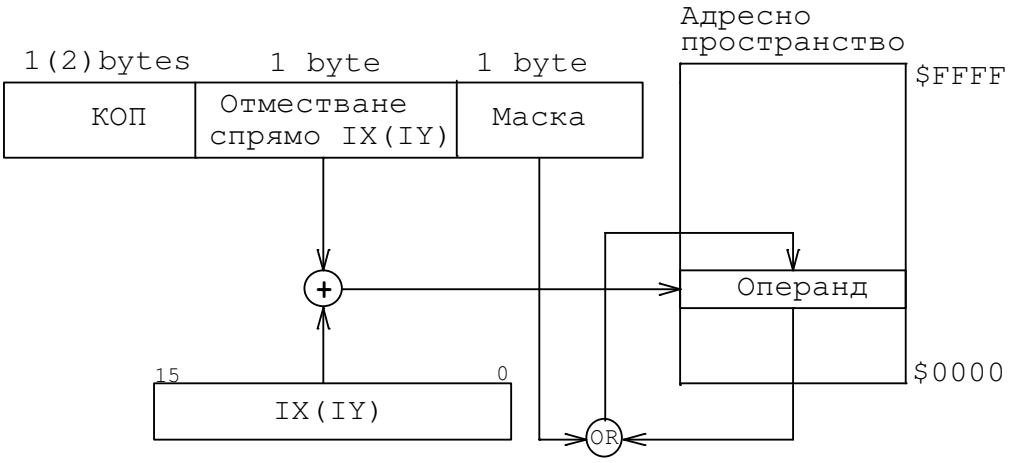
Реализират се от две машинни команди: **BSET** и **BCLR.**

- **BSET** – установяване бит в „1". Представена е на фиг.11.

Полето **„Маска"** определя кои битове на операнда ще бъдат установени в състояние „1".



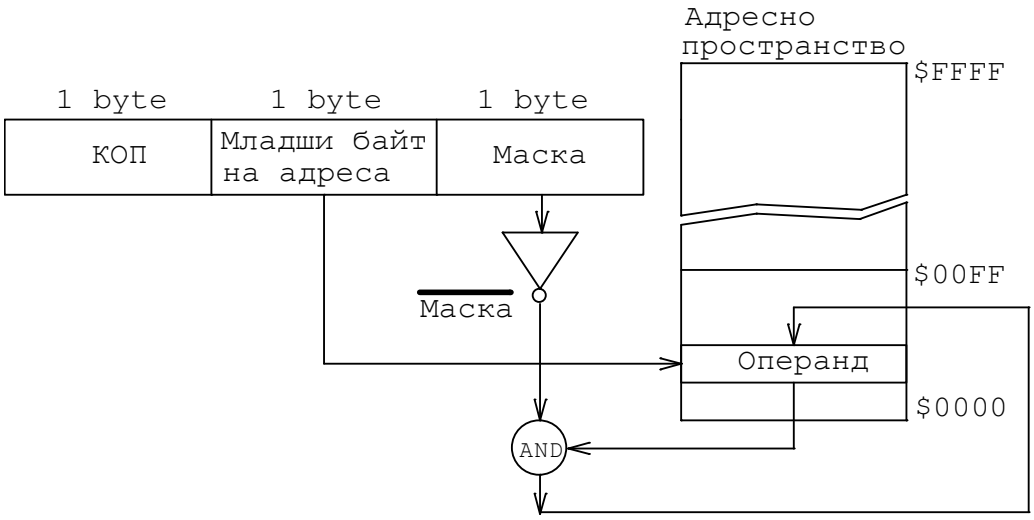
Фиг. 11А. Команда BSET с директна адресация.



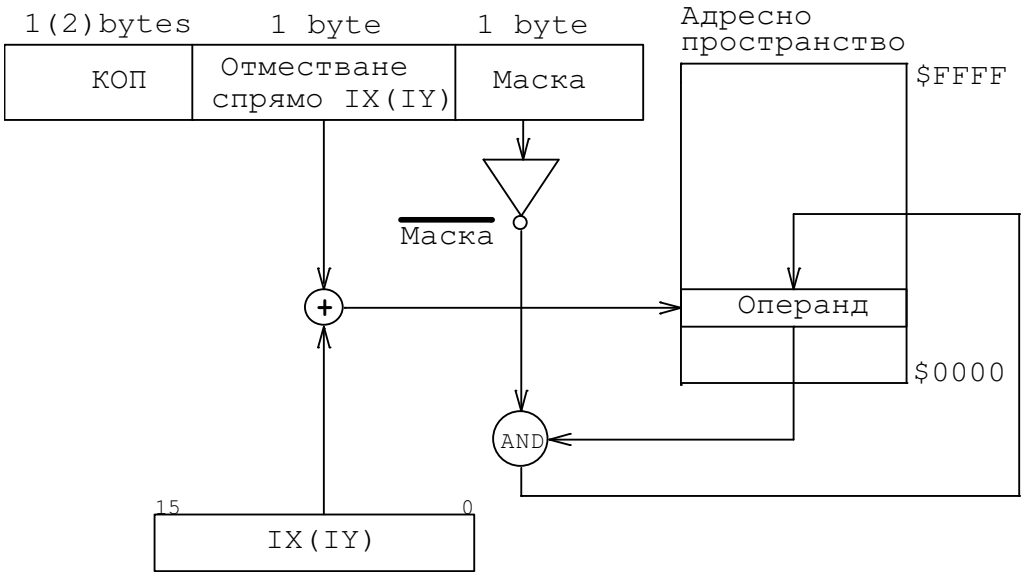
Фиг. 11Б. Команда BSET с индексна адресация.

- BCLR – нулиране на бит. Представена е на фиг.12.

Полето **„Маска”** определя кои битове на операнда ще бъдат нулирани.



Фиг. 12А. Команда BCLR с директна адресация.

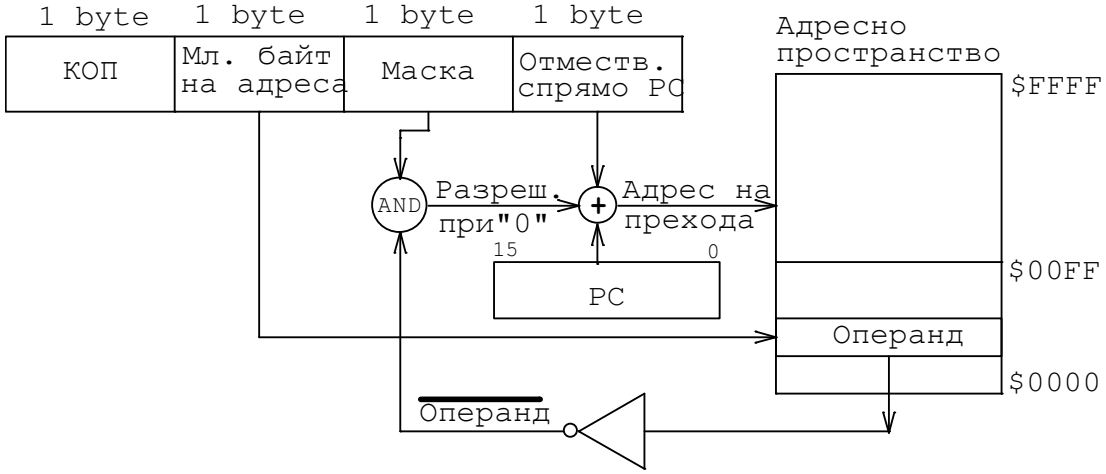


Фиг. 12Б. Команда BCLR с индексна адресация.

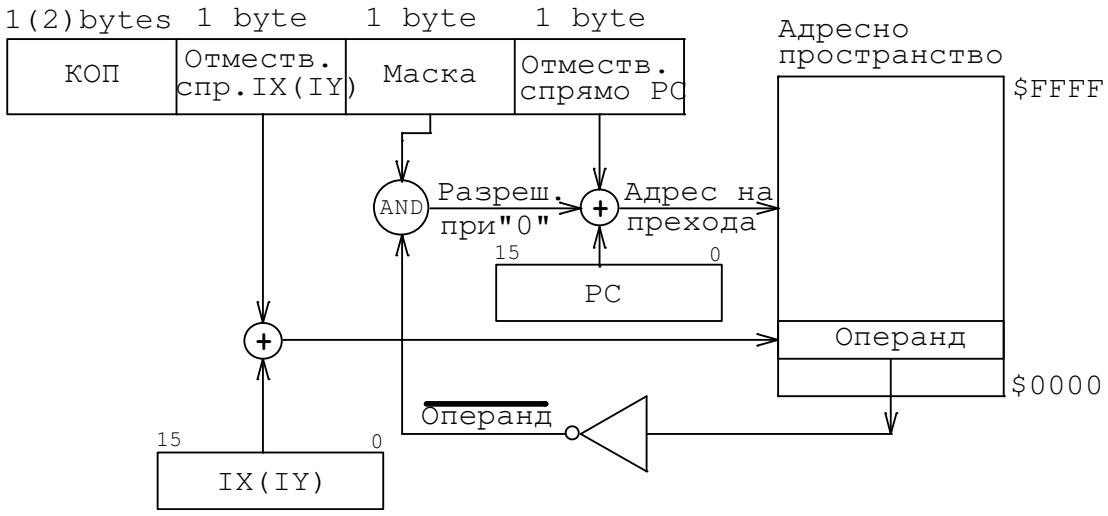
II.7.Б Операции за условен преход в зависимост от състоянието на бит.

Реализират се от две машинни команди: **BRSET** и **BRCLR.**

- **BRSET** –. Представена е на фиг.13. Полето „МАСКА" определя кои битове от операнда ще бъдат проверявани. Ако всички битове на проверявани битове на операнда съдържат „1", се извършва преход на указания адрес. В противен случай се изпълнява следващата по ред машинна команда.

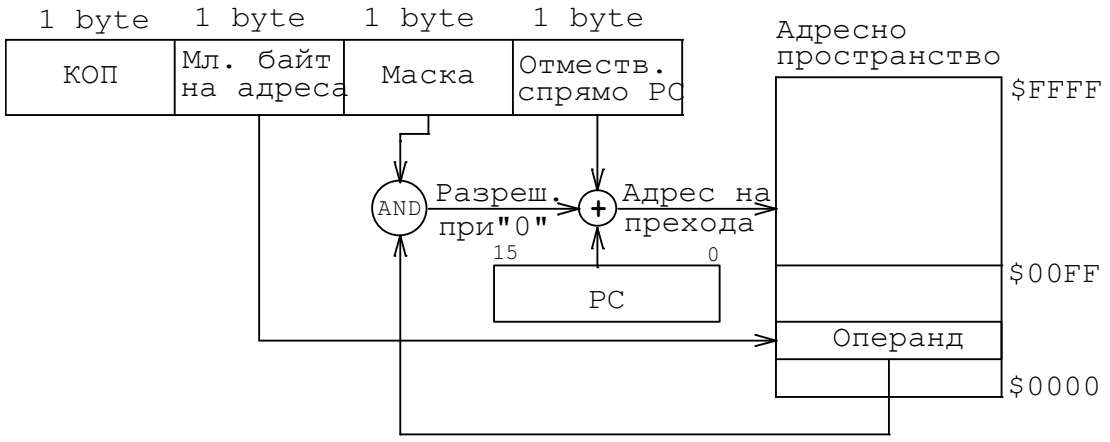


Фиг. 13А. Команда BRSET с директна адресация.

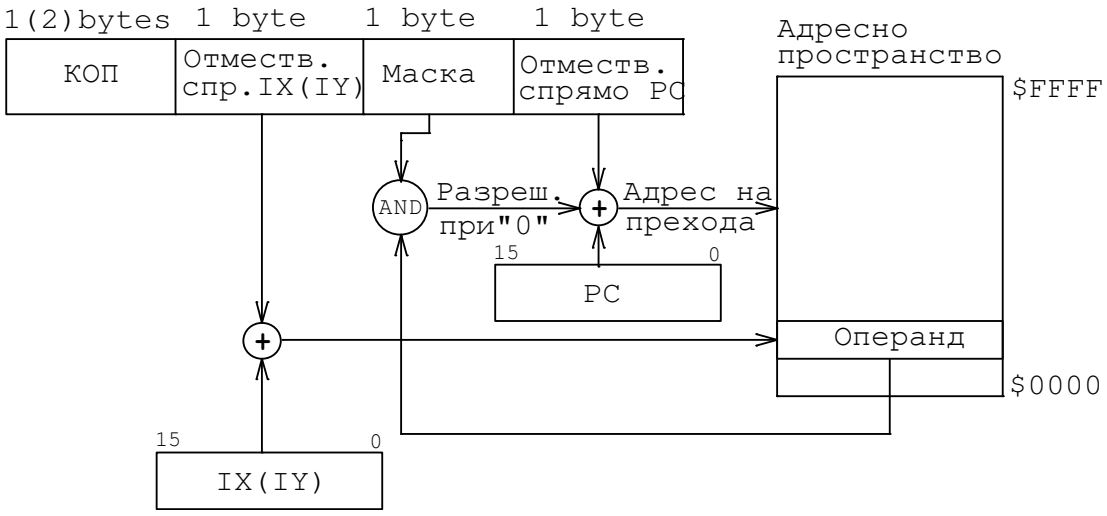


Фиг. 13Б. Команда BRSET с индексна адресация.

- **BRCLR** –. Представена е на фиг.14. Полето „МАСКА” определя кои битове от операнда ще бъдат проверявани. Ако всички битове на проверявани битове на операнда съдържат „0”, се извършва преход на указания адрес. В противен случай се изпълнява следващата по ред машинна команда.



Фиг. 14А. Команда BRCLR с директна адресация.



Фиг. 14B. Команда BRCLR с индексна адресация.