Adresní mody procesoru

Obecně o adresování

- Různé typy procesorů mohou mít v instrukci 1, 2 nebo více adres. Operandy mohou ležet v registrech nebo v paměti.
- Adresní mechanismus procesoru musí umožnit:
  - adresování instrukcí (skoky, větvení programu, podprogramy),
  - adresování jednoduchých proměnných různé délky a typu,
  - práci s indexovanými proměnnými,
  - práci se strukturami.
- Lze řešit s různým podílem HW ↔ SW.
- „Pokročilé“ požadavky:
  - segmentování programu,
  - stránkování,
  - ... .

Základní typy operandů a adres v instrukci

Podle typu procesoru lze v instrukcích používat různé typy operandů:

1. Implicitní operand
2. Registrový operand
3. Přímý operand
4. Přímá adresa
5. Nepřímá adresa
6. Nepřímá adresa v registru
7. Indexová adresa
8. Bázová adresa
9. Složená (segmentová) adresa
10. Relativní adresa

Implicitní operand

Operand je určen přímo typem instrukce.

Př.: **RTS**

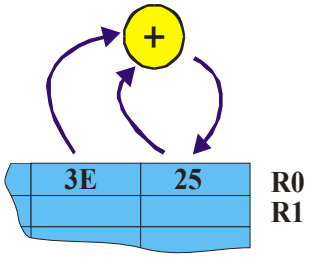
Naplní PC obsahem adresy určené SP.
PC ani SP není v instrukci explicitně uveden.

Registrový operand

Operand je v registru (registrech), které jsou v instrukci explicitně uvedeny.

Př.: **`ADD.B R0H,R0L`**

Sečte obsah R0L a R0H, výsledek je v R0L.



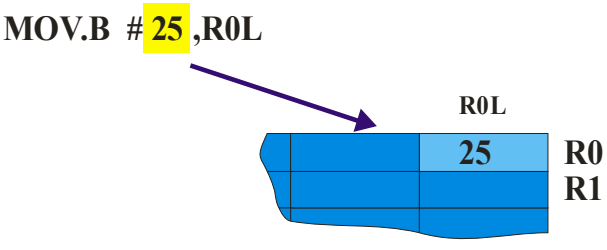
Přímý operand

Operand je přímo uveden v instrukci.

Znak # označuje přímý operand. Jeho hodnota je uložena v instrukci.

Př.: **`MOV.B #25,R0L`**

Uloží do R0L hodnotu 25 .



Přímá adresa

Operand je uložen v paměti. Jeho adresa je uvedena v instrukci

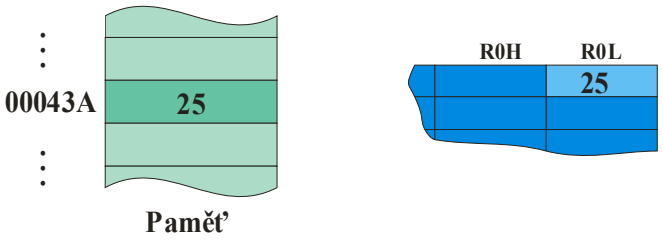
Znak @ znamená, že se jedná o adresu operandu.

Př.: **`MOV.B @043A,R0L`**

Uloží do R0L obsah adresy 043A.

Použití:

- Práce se skalární proměnnou



Uložení operandů v instrukci

- Jednotlivé typy operandů vyžadují různou délku pole pro jejich uložení v instrukci (viz dále kódování instrukcí).

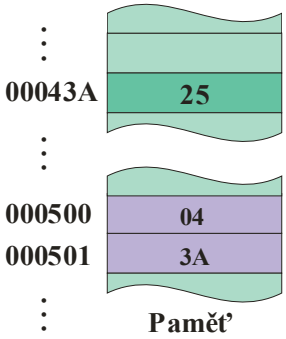
Nepřímá adresa

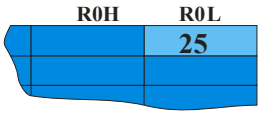
V instrukci je uvedena adresa paměti, kde je uložena adresa operandu.

Znaky @@ znamenají, že se jedná o adresu adresy operandu.

Př.: **MOV.B @@0500,R0L**

Uloží do R0L obsah adresy 043A. (H8S umí pracovat s tímto typem adres jen omezeně).





Použití:

- Implementace pointeru

Nepřímá adresa v registru

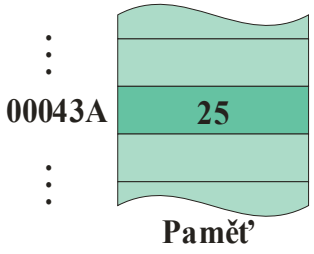
V instrukci je uvedena adresa registru, kde je uložena adresa operandu.

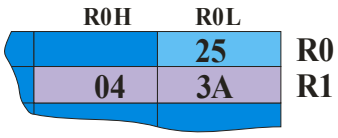
Př.: **MOV.B @R1,R0L**

Uloží do R0L obsah adresy 043A.

Použití:

- Implementace pointeru



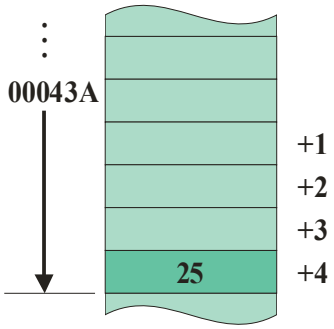


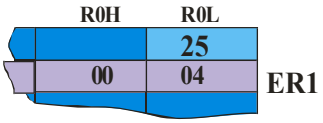
Indexová adresa

Poloha operandu v paměti je určena součtem adresy uvedené v instrukci a obsahem indexového registru.

Př.: **`MOV.B @(043A+ER1),R0L`**

Uloží do R0L obsah adresy 043A + 4.





Použití:

- Práce s indexovanou proměnnou (pole).

Bázová adresa

Poloha operandu v paměti je určena součtem obsahu registru a offsetu uvedeného v instrukci.

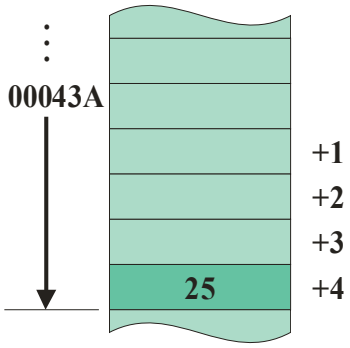
Použití:

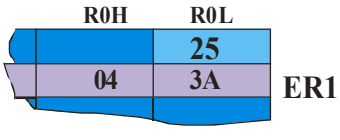
- Velké adresní prostory.
- Práce se strukturami.

Př.: **`MOV.B @(4+ER1),R0L`**

Uloží do R0L obsah adresy 043A + 4.

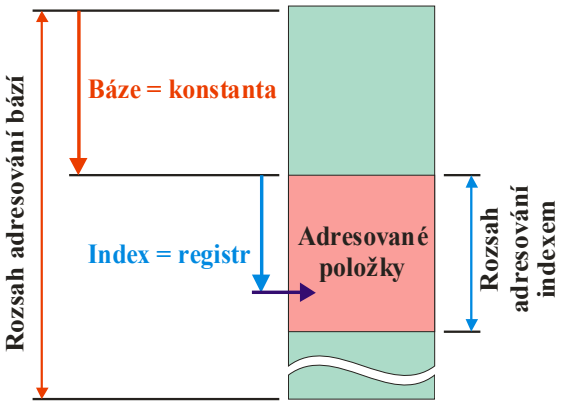
Implementace bázové a indexové adresy může ale nemusí být stejná.



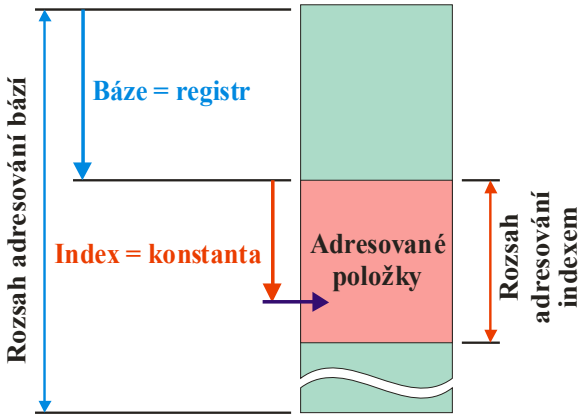


Indexová a bázová adresa (1)

```
MOV.B   @(043A+ER1),R0L
```



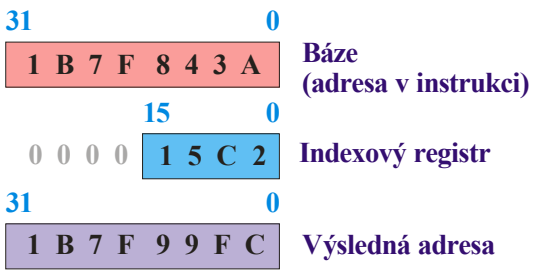
Indexová adresa



Bázová adresa

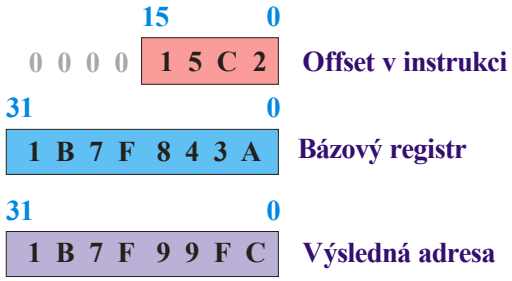
Indexová a bázová adresa (2)

- Báze = 32 bitů ⇒ 4 GB
- Index = 16 bitů ⇒ 64 kB



Indexová adresa

- Offset = 16 bitů ⇒ 64 kB
- Báze = 32 bitů ⇒ 4 GB



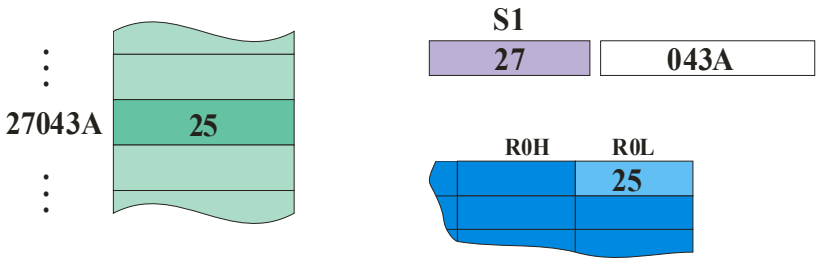
Bázová adresa

Složená (segmentová) adresa

Adresa operandu v paměti se vypočítá složením z několika částí, uložených v registrech nebo v operandovém poli instrukce.

Př.: **MOV.B @(S1:043A),R0L**

Uloží do R0L obsah adresy určené složením registru S1 a hodnoty 043A z operandového pole instrukce. **Tento typ instrukce (ani registr S1) není v instrukčním souboru procesoru H8S.**

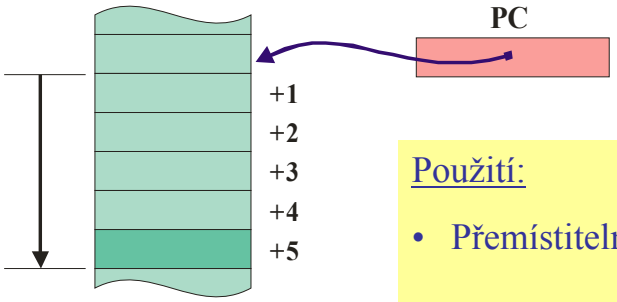


Relativní adresa

Adresa operandu v paměti se určí jako součet obsahu PC a offsetu, uloženého v operandovém poli instrukce nebo v některém registru.

Př.: **BCS 0005**

Provede se skok na adresu PC + 5 (je-li splněna podmínka C = 1). Před zvětšením o 5 ukazuje PC na adresu **za** instrukcí BCS.



Použití:

- Přemístitelný program.

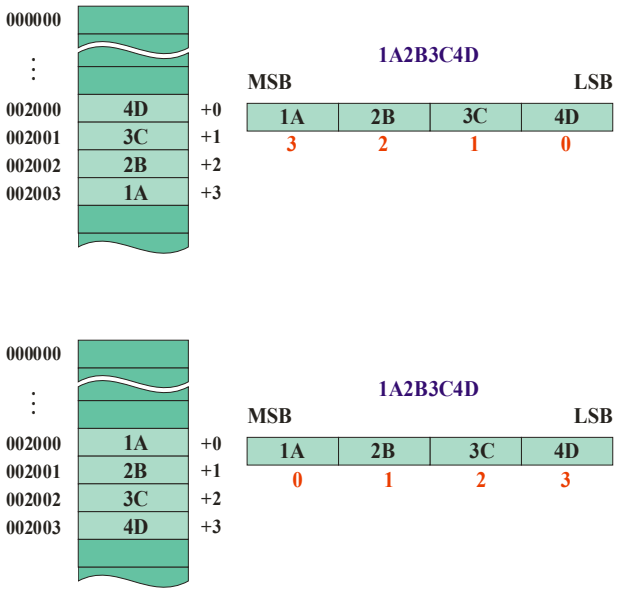
Little a Big Endian

Data delší než 1 byte mohou být do paměti ukládána v pořadí

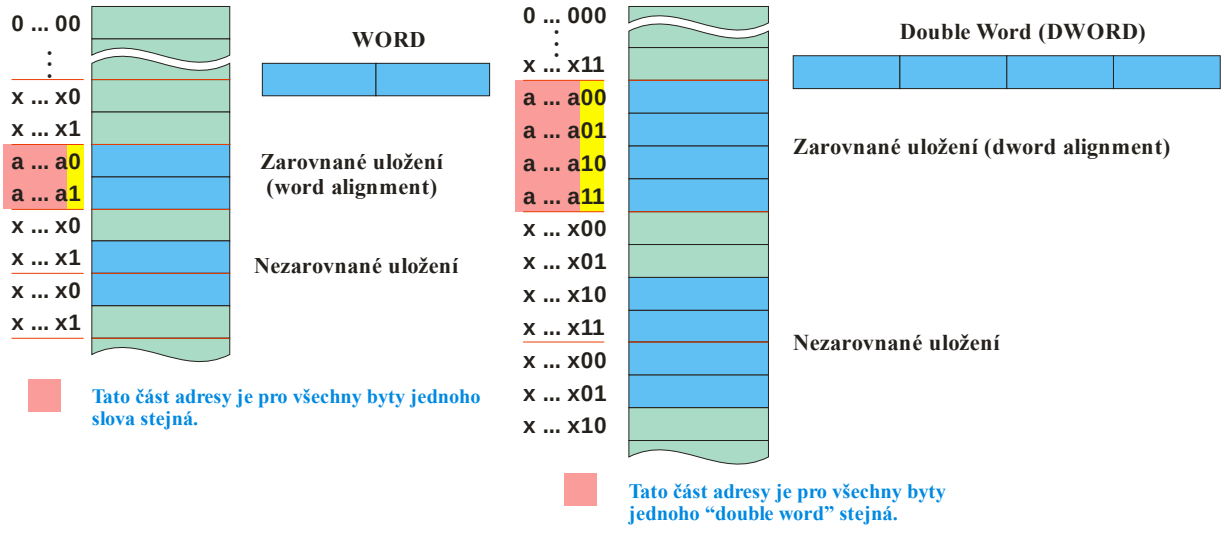
- **LSB ... MSB (= Little Endian),**
- **MSB ... LSB (= Big Endian).**

Intel IA-32 je Little Endian.

H8S je Big Endian.

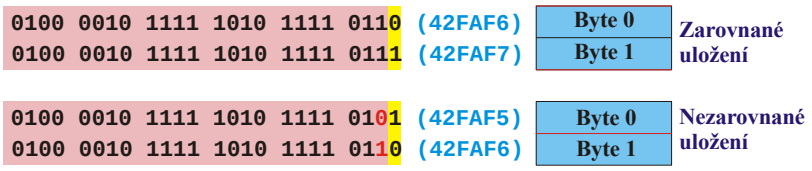


Zarovnání dat (Alignment) (1)



Zarovnání dat (Alignment) (2)

- Zarovnané a nezarovnané uložení Word:



- Zarovnané a nezarovnané uložení Double Word: