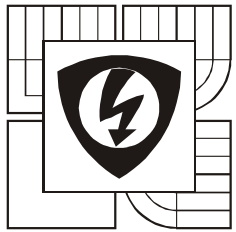
Možnosti adresování



Kurz: MSPR – Signálové procesory

Autor: Petr Sysel

Lektor: Petr Sysel

Možnosti zadání dat

- Bezprostřední data
  - data se stávají součástí instrukce nebo následují bezprostředně za ní,
  - v průběhu programu je již nelze měnit,
  - Příklad: `move.w #50,X0`
    `asll.w #7,A`
- Přímá adresace
  - součástí instrukce je adresa paměti, kde jsou data uložena,
  - Příklad: `move.l X:$5420,Y`
    `add.l X:$7f6e,A`
- Nepřímá adresace
  - součástí instrukce je předpis, jak získat adresu dat v paměti,
  - Příklad: `move.l X:(R2),A`
    `add.w X:(R0+$0DC5),B`

Specifika řady TMS320C6000

- Přímá data jsou omezena do velikosti 16 bitů – stávají se součástí strojového kódu instrukce,
- přímá data nemohou být bezprostředně za instrukcí,
- přímá adresace v podstatě není možná – adresa je 32bitová,
- pro nepřímou adresaci je možné použít libovolný registr,
- nepřímou adresaci provádí jednotka D.

Adresovací instrukce

- Pro čtení z paměti slouží sada instrukcí `LDx` – LoaD,
- pro zápis do paměti slouží sada instrukcí `STx` – STore,
- liší se v délce přenášených dat a zpracování znaménka.

délka dat	datový typ	se znaménkem	bez znaménka	pomůcka
8 bitů	`char`	`LDB/STB`	`LDBU`	`Byte`
16 bitů	`short`	`LDH/STH`	`LDHU`	`Half word`
32 bitů	`int`	`LHW/STW`		`Word`
64 bitů*	`long long`	`LDDW/STDW`		`Double Word`

*pouze TMS320C6400

Možnosti adresování

- Hodnotu adresovacího registru lze změnit před nebo po adresaci:

adresace	beze změny no modification	změna před pre-inc/dec-rement	změna po post-inc/dec-rement
nepřímá	`*R`	`*++R` `*--R`	`*R++` `*R--`
relativní	`*+R[ucst5]` `*-R[ucst5]`	`*++R[ucst5]` `*--R[ucst5]`	`*R[ucst5]++` `*R[ucst5]--`
indexová	`*+R[offsetR]` `*-R[offsetR]`	`*++R[offsetR]` `*--R[offsetR]`	`*R[offsetR]++` `*R[offsetR]--`
relativní	`*+B14[ucst15]` `*+B15[ucst15]`		

- `ucst5` – neznaménková 5bitová konstanta,
- `ucst15` – neznaménková 15bitová konstanta.

Změna adresy bez přenosu

- Offset udává počet prvků dané velikosti – před přičtením k registru se posune vlevo o počet bajtů přenášených dat:
  - `LDB *++A4[3],A7` – hodnota registru A4 se před adresací zvýší o 3,
  - `LDW *++A4[3],A7` – hodnota registru A4 se před adresací zvýší o 12 = 3 * 4 bajty.
- pro změnu adresy bez přenosu lze použít sadu instrukcí `ADDAx` a `SUBAx`,
- druhý operand je posunut vlevo o tolik bitů, kolik bajtů obsahují data.

délka dat	datový typ	zvýšení	snížení
8 bitů	`char`	`ADDAB`	`SUBAB`
16 bitů	`short`	`ADDAH`	`SUBAH`
32 bitů	`int`	`ADDAW`	`SUBAW`
64 bitů*	`long long`	`ADDAD`	

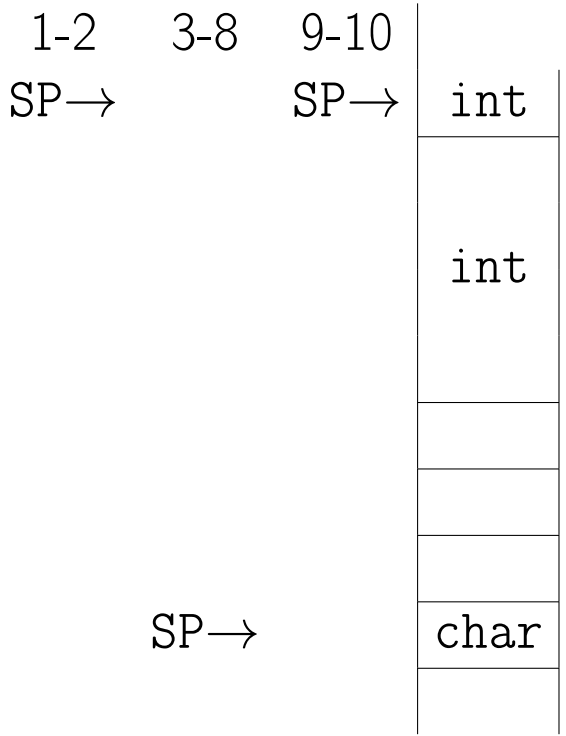
*pouze TMS320C6400

Využití změny adresy

- Relativní lze použít pro adresování struktur (objektů):
  - registr je nastaven na adresu struktury,
  - konstanta ucst5 je použita jako offset pro jednotlivé prvky,
  - nevýhoda je malý rozsah.
- relativní s registrem B14/B15 lze použít pro realizaci zásobníku nebo haldy:
  - registr je nastaven na adresu vrcholu zásobníku,
  - konstanta `ucst15` se použije jako offset pro jednotlivé prvky na zásobníku,
  - nevýhoda v omezeném počtu možných registrů,
  - při použití jako ukazatel zásobníku/haldy je nutné dodržet:
    - symetrická práce – před návratem z funkce musí ukazatel zásobníku mít stejnou hodnotu jako při vstupu do funkce,
    - zarovnání – ukazatel zásobníku musí být zarovnán na největší adresovatelná data – 8 bajtů (používá se napříč programem pro přístup k různě velkým datům),
  - v prostředí je nastaveno synonymum DP↔B14 (Data Page pointer), SP↔SP (Stack Pointer).

Příklad využití zásobníku

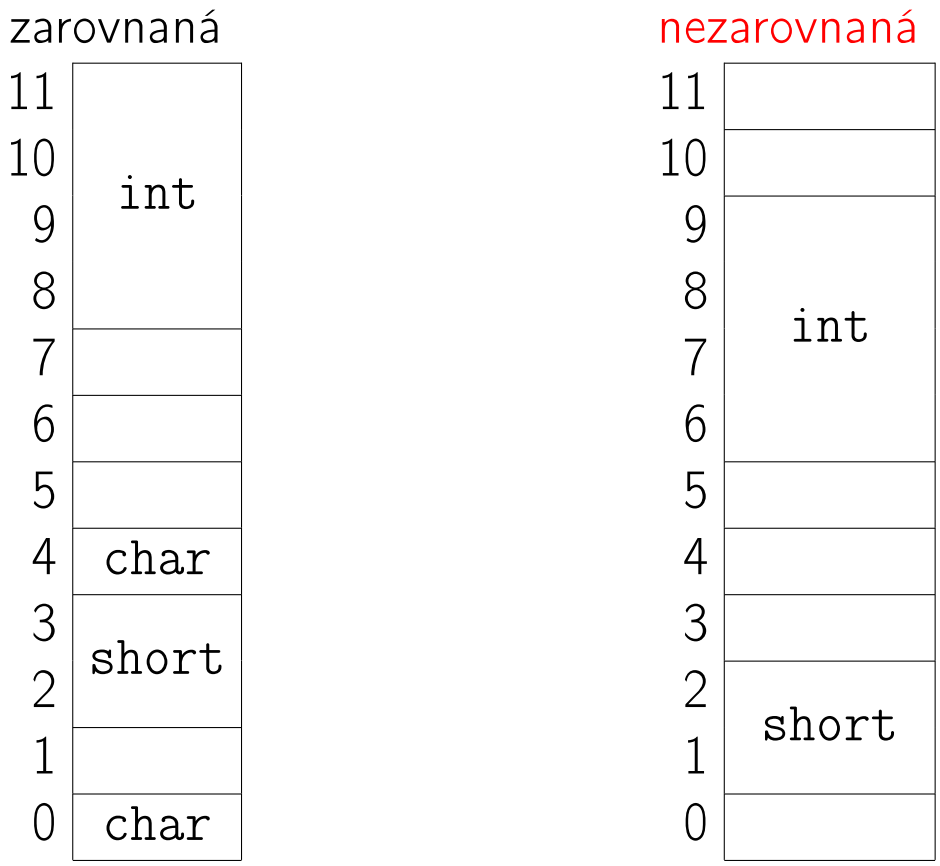
```
1  _func:
2      SUB     SP,8,SP      ; vytvoření místa
3
4      LDW     *+SP[1],A6   ; přístup k int
5
6      LDB     *+SP[0],B5   ; přístup k char
7
8      ADD     SP,8,SP      ; uvolnění místa
9
10     B       B3           ; návrat z funkce
```



Zarovnání dat

- Při adresování dat velikosti K bajtů je $K - 1$ nejméně významných bitů nulovaných,
- instrukce `LDx`/`STx` tyto bity nulují automaticky,
- hodnoty musí být v paměti zarovnány (alignement):
  - adresa musí být celočíselným násobkem počtu bajtů dat,
  - `char` může být uložen na libovolné adrese,
  - `int` musí být na adrese, která je násobkem 4.
- překladač data zarovnává automaticky,
- problém může nastat při ruční adresaci nebo umístění dat,
- největší problém při zápisu – hrozí přepsání okolních hodnot,
- u řady TMS320C6400 existují instrukce `LDNx` a `STNx` pro nezarovnaná data – trvají však 2x déle.

Zarovnání dat



Důsledek nezarovnání dat

	před
7	0x02
6	0x46
5	0x8A
4	0x00
3	0xFF
2	0xCE
1	0xFF
0	0xFF

```
MVKL 0x00000006,A4
MVKL 0x00000006,A4 ; A4 = 0x00000006

LDW *A4--,A5       ; A4 = 0x00000002
NOP     4          ; A5 = 0x02468A00

LDB *++A4[1],A6    ; A4 = 0x00000003
OR A6,A5,A6        ; A6 = 0x02468ACE

STW A6,*A4
```

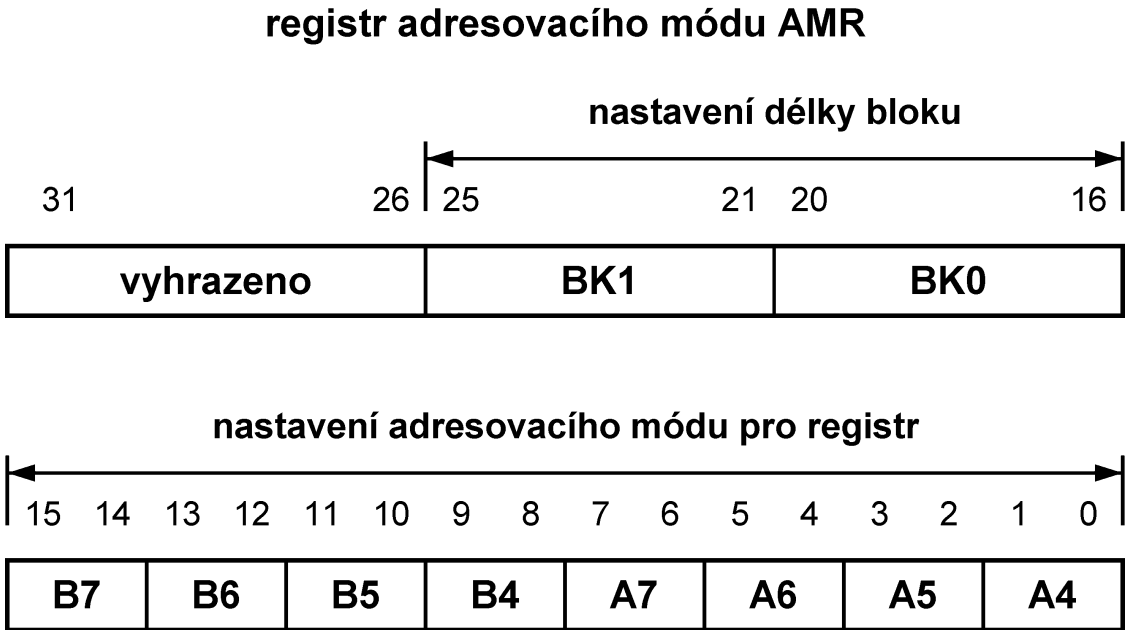
	po
7	0x02
6	0x46
5	0x8A
4	0x00
3	0x02
2	0x46
1	0x8A
0	0xCE

Adresovací režimy

- Lineární adresování:
  - adresa se zvyšuje až k nejvyšší adrese,
  - standarní režim.
- Modulo adresování (circullar):
  - adresa se pohybuje v rámci vyhrazeného bloku,
  - režim je možný u většiny signálových procesorů.
- Bitově-reverzní adresování:
  - adresa je vytvořena opačným pořadím bitů (LSB->MSB),
    $(0)_{10} = (00)_2 \leftrightarrow (00)_2 = (0)_{10}$
    $(1)_{10} = (01)_2 \leftrightarrow (10)_2 = (2)_{10}$
    $(2)_{10} = (10)_2 \leftrightarrow (01)_2 = (1)_{10}$
    $(3)_{10} = (11)_2 \leftrightarrow (11)_2 = (3)_{10}$
  - u některých signálových procesorů pro podporu rychlé Fourierovy transformace.

Modulo adresování u TMS320C6000

- Nastavení v řídicím registru AMR – Addressing Mode Register.



Adressing Mode Register

- `BK0`, `BK1` – nastavení délky modulo adresování, délka modulo bloku je rovna celočíselné mocnině 2: $N = 2^{(K+1)}$, kde K je hodnota v `BKx`,

Adressing Mode Register

- `BK0`, `BK1` – nastavení délky modulo adresování, délka modulo bloku je rovna celočíselné mocnině 2: $N = 2^{(K+1)}$, kde K je hodnota v `BKx`,
- `B7`, `B6`, `B5`, `B4`, `A7`, `A6`, `A5`, `A4` přiřazení délky bloku adresovacímu registru:

Adressing Mode Register

- BK0, BK1 – nastavení délky modulo adresování, délka modulo bloku je rovna celočíselné mocnině 2: $N = 2^{(K+1)}$, kde K je hodnota v BKx,
- B7, B6, B5, B4, A7, A6, A5, A4 přiřazení délky bloku adresovacímu registru:
  - 00 – lineární adresování,

Adressing Mode Register

- BK0, BK1 – nastavení délky modulo adresování, délka modulo bloku je rovna celočíselné mocnině 2: $N = 2^{(K+1)}$, kde K je hodnota v BKx,
- B7, B6, B5, B4, A7, A6, A5, A4 přiřazení délky bloku adresovacímu registru:
  - 00 – lineární adresování,
  - 01 – modulo adresování s délkou podle bloku BK0,

Adressing Mode Register

- BK0, BK1 – nastavení délky modulo adresování, délka modulo bloku je rovna celočíselné mocnině 2: $N = 2^{(K+1)}$, kde K je hodnota v BKx,
- B7, B6, B5, B4, A7, A6, A5, A4 přiřazení délky bloku adresovacímu registru:
  - 00 – lineární adresování,
  - 01 – modulo adresování s délkou podle bloku BK0,
  - 10 – modulo adresování s délkou podle bloku BK1,

Adressing Mode Register

- BK0, BK1 – nastavení délky modulo adresování, délka modulo bloku je rovna celočíselné mocnině 2: $N = 2^{(K+1)}$, kde K je hodnota v BKx,
- B7, B6, B5, B4, A7, A6, A5, A4 přiřazení délky bloku adresovacímu registru:
  - 00 – lineární adresování,
  - 01 – modulo adresování s délkou podle bloku BK0,
  - 10 – modulo adresování s délkou podle bloku BK1,
  - 11 – reservované pro další použití.

Adressing Mode Register

- BK0, BK1 – nastavení délky modulo adresování, délka modulo bloku je rovna celočíselné mocnině 2: $N = 2^{(K+1)}$, kde K je hodnota v BKx,
- B7, B6, B5, B4, A7, A6, A5, A4 přiřazení délky bloku adresovacímu registru:
  - 00 – lineární adresování,
  - 01 – modulo adresování s délkou podle bloku BK0,
  - 10 – modulo adresování s délkou podle bloku BK1,
  - 11 – reservované pro další použití.
- u ostatních registrů je vždy lineární adresování,

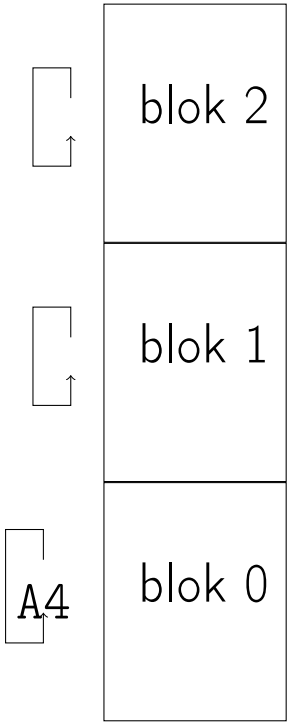
Adressing Mode Register

- BK0, BK1 – nastavení délky modulo adresování, délka modulo bloku je rovna celočíselné mocnině 2: $N = 2^{(K+1)}$, kde K je hodnota v BKx,
- B7, B6, B5, B4, A7, A6, A5, A4 přiřazení délky bloku adresovacímu registru:
  - 00 – lineární adresování,
  - 01 – modulo adresování s délkou podle bloku BK0,
  - 10 – modulo adresování s délkou podle bloku BK1,
  - 11 – reservované pro další použití.
- u ostatních registrů je vždy lineární adresování,
- modulo adresování znamená, že v paměti je vyhrazen blok dané délky a adresa jej nemůže opustit - pohybuje se v kruhu (circular buffer),

Adressing Mode Register

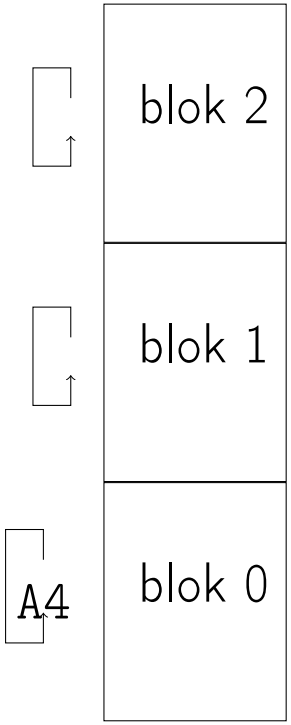
- BK0, BK1 – nastavení délky modulo adresování, délka modulo bloku je rovna celočíselné mocnině 2: $N = 2^{(K+1)}$, kde K je hodnota v BKx,
- B7, B6, B5, B4, A7, A6, A5, A4 přiřazení délky bloku adresovacímu registru:
  - 00 – lineární adresování,
  - 01 – modulo adresování s délkou podle bloku BK0,
  - 10 – modulo adresování s délkou podle bloku BK1,
  - 11 – reservované pro další použití.
- u ostatních registrů je vždy lineární adresování,
- modulo adresování znamená, že v paměti je vyhrazen blok dané délky a adresa jej nemůže opustit - pohybuje se v kruhu (circular buffer),
- bloků je v paměti více – konkrétní je vybrán počáteční inicializací adresovacího registru.

Modulo adresování



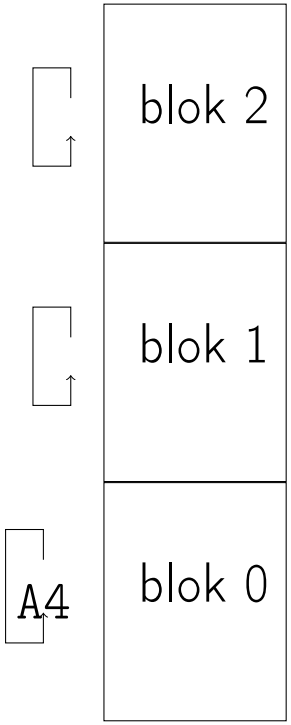
- Délka buferu musí být mocnina 2
  $N = 2^K, K \in \Re,$

Modulo adresování



- Délka buferu musí být mocnina 2 $N = 2^K, K \in \Re$,
- adresa začátku buferu musí mít K nejnižších bitů nulových, tj. musí být násobkem N,

Modulo adresování



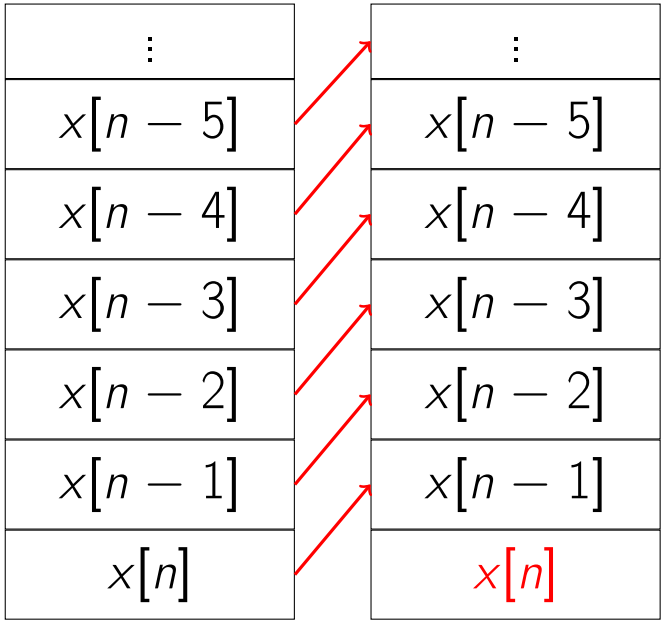
- Délka buferu musí být mocnina 2 $N = 2^K, K \in \Re$,
- adresa začátku buferu musí mít K nejnižších bitů nulových, tj. musí být násobkem N,
- začátek je tedy zarovnán na velikost bloku.

Využití kruhového adresování

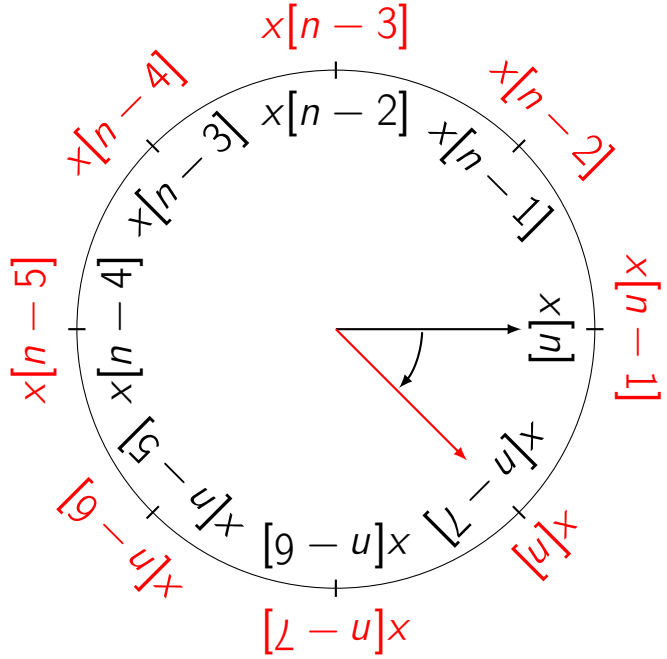
- Plovoucí zpoždění:
  - nové vzorky zapisujeme pomocí jedné adresy,
  - zpožděné vzorky čteme z jiné posunuté adresy.
- Vyrovnávací paměť:
  - zdroj zapisuje do kruhové paměti na jednu adresu,
  - cíl vyčítá z kruhové paměti ze stejné nebo jiné adresy.

Realizace zpoždění

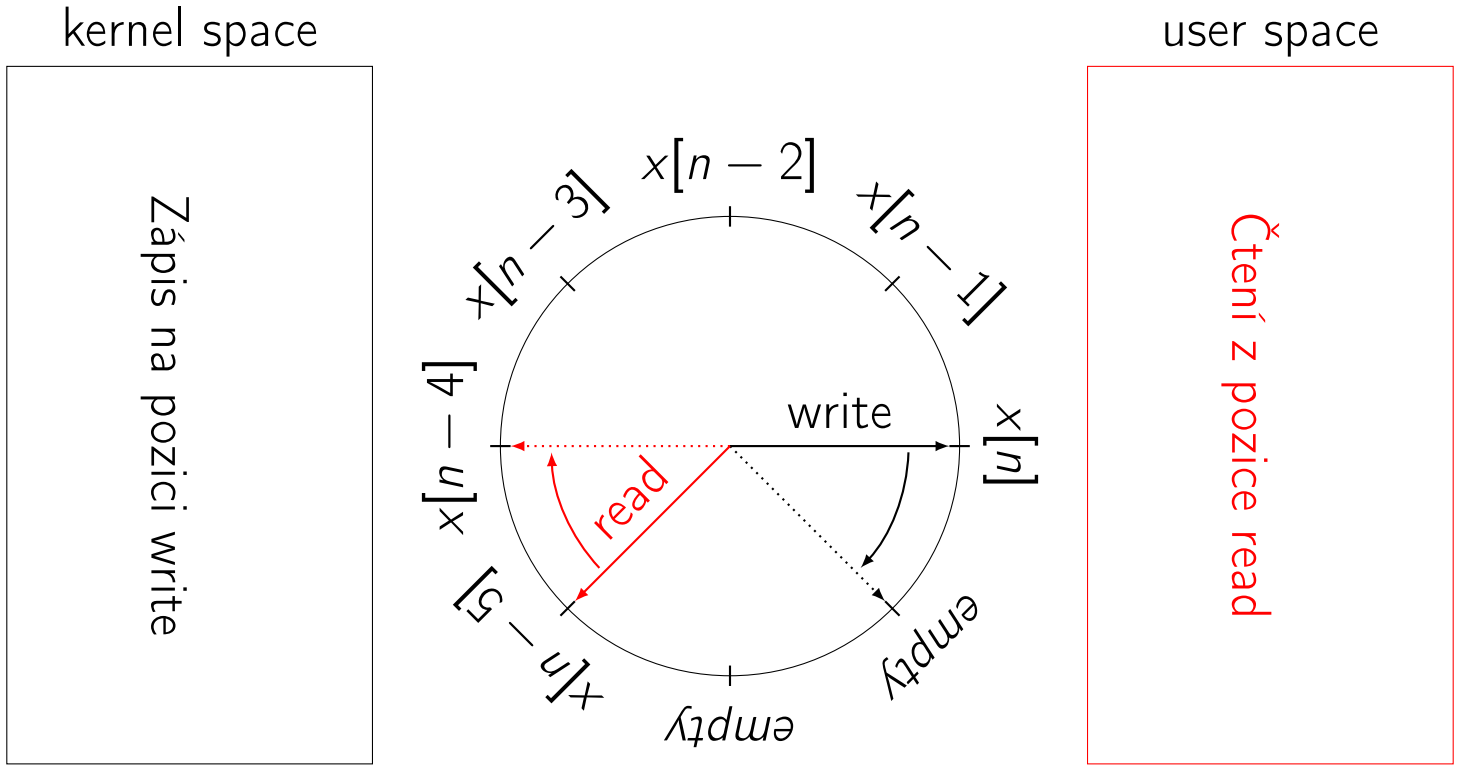
lineární adresování



modulo adresování



Realizace vyrovnávací paměti



Kruhová paměť bez hardwarové podpory

- Pomocí operace **mod**$(ind, delka)$ – zbytek po dělení:
  - \+ délka může být libovolná,
  - – operace je velmi výpočetně náročná.
- pomocí bitového součinu $ind\&(delka-1)$:
  - \+ výpočetně velice jednoduchá operace,
  - – funguje pouze pro délku rovnou mocnině 2: $delka = 2^K, K \in \mathcal{N}$,
  - příklad pro $K = 2, delka = 2^2 = 4, delka - 1 = 3 = (0\ldots011)_2$

$$
\begin{aligned}
(0)_{10} &= (0\ldots0000)_2 \,\&\, (0\ldots011)_2 = (0\ldots0000)_2 = (0)_{10}\\
(1)_{10} &= (0\ldots0001)_2 \,\&\, (0\ldots011)_2 = (0\ldots0001)_2 = (1)_{10}\\
(2)_{10} &= (0\ldots0010)_2 \,\&\, (0\ldots011)_2 = (0\ldots0010)_2 = (2)_{10}\\
(3)_{10} &= (0\ldots0011)_2 \,\&\, (0\ldots011)_2 = (0\ldots0011)_2 = (3)_{10}\\
(4)_{10} &= (0\ldots0100)_2 \,\&\, (0\ldots011)_2 = (0\ldots0000)_2 = (0)_{10}\\
(5)_{10} &= (0\ldots0101)_2 \,\&\, (0\ldots011)_2 = (0\ldots0001)_2 = (1)_{10}
\end{aligned}
$$

Organizace paměti

- Nejmenší adresovatelná položka v paměti je 1 bajt,
- více bajtové hodnoty mohou být v paměti uloženy dvěma způsoby:
  little endian na nižší adresy jsou ukládány nižší bajty,
  big endian na nižší adresy jsou ukládány horní bajty.
- způsoby nelze vzájemně kombinovat.