Rozšiřování hodnot, sčítání, odčítání, posuvy, rotace, násobení a dělení

ISU-cv04

Ing. Jakub Husa Ph.D.

Vysoké učení technické v Brně, Fakulta informačních technologií
Božetěchova 1/2. 612 66 Brno - Královo Pole

ihusa@fit.vut.cz

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

7. března 2025

Rozšiřování hodnot

Rozšiřování hodnot

Sčítat a odčítat můžeme pouze hodnoty stejné velikosti (datového typu):

- Pokud se velikost hodnot liší, tu menší z nich musíme rozšířit (extend) tak, aby se prodloužil jeho zápis (8b->16b->32b), ale nezměnila se jeho hodnota.

Kladná čísla se prodlužují nulami, záporná čísla se prodlužují jedničkami:

```
                ;|      AX = ????    |
mov   al,   10  ;|????????|00001010|
                ;|AH = ???|AL =   10|
                ;|-----------------|
                ;|      AX =   10    |
mov   ah,    0  ;|00000000|00001010|
                ;|AH =    0|AL =   10|
```

```
                ;|      BX = ????    |
mov   bl, -10   ;|????????|11110110|
                ;|BH = ???|BL = -10|
                ;|-----------------|
                ;|      BX =  -10    |
mov   bh,  -1   ;|11111111|11110110|
                ;|BH =   -1|BL = -10|
```

Pokud číslo prodloužíme špatně, jeho hodnota se změní:

```
                ;|      CX = ????    |
mov   cl,   10  ;|????????|00001010|
                ;|CH = ???|CL =   10|
                ;|-----------------|
                ;|      CX = -254    |
mov   ch,  -1   ;|11111111|00001010|
                ;|CH =   -1|CL =   10|
```

```
                ;|      DX = ????    |
mov   dl, -10   ;|????????|11110110|
                ;|DH = ???|DL = -10|
                ;|-----------------|
                ;|      DX =  246    |
mov   dh,    0  ;|00000000|11110110|
                ;|DH =    0|DL = -10|
```

Instrukce rozšíření

Pro rozšiřování čísel používáme instrukce MOVZX a MOVSX:

- Instrukce se chovají jako MOV, ale jejich cílový operand je větší než zdrojový.
- MOVZX (MOVe Zero eXtend) prodlužuje nulami (bez znaménka).
- MOVSX (MOVe Sign eXtend) prodlužuje nejlevějším bitem (se znaménkem).

```
movzx  ax, al ;  AX = AL       (bez-znamenkove rozsireni z  8b na 16b)
movzx eax, ax ; EAX = AX       (bez-znamenkove rozsireni z 16b na 32b)
movzx eax, al ; EAX = AL       (bez-znamenkove rozsireni z  8b na 32b)
```

Pro rozšiřování se znaménkem existují také instrukce CBW, CWD, CDQ a CWDE:

- Tyto instrukce nemají operandy, a budeme je používat až při násobení a dělení.

```
cbw            ;    AX  =  AL   (Convert  Byte to        Word)
cwd            ; DX:AX  =  AX   (Convert  Word to Double-word)
cdq            ;EDX:EAX = EAX   (Convert Dword to   Quad-word)
cwde           ;    EAX =  AX   (Convert  Word to Double-word Extended)
```

- Dvojtečka znamená že velké číslo je rozděleno do dvou menších registrů:

  DX:AX je 32b číslo které má horních 16b v DX, a spodních 16b v AX.

  EDX:EAX je 64b číslo které má horních 32b v EDX, a spodních 32b v EAX.

Na vyzkoušení

Zdrojovým (ale ne cílovým) operandem MOVZX a MOVSX může být i proměnná:

```
%include 'rw32.inc'            ;knihovna pro vstup a vystup

section .data                  ;inicializovany segment
     X db 0b11001000           ; 8b promenna X s hodnotu 200 a/nebo -56

section .text                  ;kodovy segment
main:
    mov   al, [X]              ; prirazeni AL = [X]
    call  WriteInt8NewLine     ;  8b vypis se znamenkem (  -56)
    call  WriteUInt8NewLine    ;  8b vypis bez znamenka (  200)

    movzx ax, [X]              ; rozsireni z 8b na 16b (bez znamenka)
    call  WriteInt16NewLine    ; 16b vypis se znamenkem (  200)
    call  WriteUInt16NewLine   ; 16b vypis bez znamenka (  200)

    movsx ax, [X]              ; rozsireni z 8b na 16b (se znamenkem)
    call  WriteInt16NewLine    ; 16b vypis se znamenkem (  -56)
    call  WriteUInt16NewLine   ; 16b vypis bez znamenka (65480)
    ret
```

Na procvičení

Vyzkoušejte si:

- Vytvořte si 16b proměnnou (X) a 8b proměnnou (Y).
- Vytvořte si 32b proměnnou (K), 16b proměnnou (L) a 8b proměnnou (M).
- Bez znaménka spočítejte X = X+Y.
- Se znaménkem spočítejte K = K+L-M.

Například:

- X = 1000 => X = 1200
  Y = 200
- K = -1000 => K = -1090
  L = -100
  M = -10

Nápověda:

- Pozor na omezení překladače – při rozšiřování do 32b registru, musíme pseudo-instrukcí určit jestli rozšiřujeme 8b nebo 16b proměnnou.
- V kódovém segmentu – byte (8b), word (16b) a dword (32b).

Sčítání a odčítání

Rozšířené sčítání a odčítaní

Výsledek součtu nebo rozdílu se do registru nemusí vejít (viz. cv01):

- To jestli se výsledek výpočtu do registru vešel (0) nebo nevešel (1) procesor automaticky nastavuje do příznaku (flag) jménem CF (Carry Flag, viz. cv05).

```
mov  al, 10 ;AL=00001010= 10, CF=?
mov  bl, 20 ;BL=00010100= 20, CF=?
add  al, bl ;AL=00011110= 30, CF=0
```

```
mov  al,100 ;AL=01100100=100, CF=?
mov  bl,200 ;BL=11001000=200, CF=?
add  al, bl ;AL=00101100= 46, CF=1
```

Kromě obyčejného sčítání ADD (ADD) a odčítání SUB (SUBtract), procesor obsahuje rozšířené instrukce které umožňují CF zahrnout do následujícího výpočtu:

- ADC (ADd with Carry) – k prvnímu operandu přičte druhý, a přičte CF.
- SBB (SuBtract with Borrow) – od prvního operandu odečte druhý, a odečte CF.

```
mov  al, -1 ; AL = 11111111 = -1,  CF=?
add  al,  1 ; AL = 00000000 =  0,  CF=1
adc  al,  1 ; AL = 00000010 =  2,  CF=0
```

Inkrementace INC (INCrement) a dekrementace DEC (DECrement) přičtou nebo odečtou jedničku, podobně jako sčítání a odčítání, ale nenastavují příznak CF:

```
mov  al,  0 ; AL = 00000000 =  0,  CF=?
inc  al     ; AL = 00000001 =  1,  CF=?
dec  al     ; AL = 00000000 =  0,  CF=?
```

Na vyzkoušení (1/2)

16b součet implementovaný pomocí dvou 8b součtů:

```
%include 'rw32.inc'          ;knihovna pro vstup a vystup
section .text                ;kodovy segment
main:
    mov ax,  1000             ; AX = 00000011 11101000 =  1000
    mov bx, -2000             ; BX = 11111000 00110000 = -2000
    add ax, bx                ; AX = 11111100 00011000 = -1000
    call WriteInt16NewLine    ; (jeden 16b soucet - spravne)

    mov ax,  1000             ; AX = 00000011 11101000 =  1000
    mov bx, -2000             ; BX = 11111000 00110000 = -2000
    add al, bl                ; AX = 00000011 00011000 =   792
    add ah, bh                ; AX = 11111011 00011000 = -1256
    call WriteInt16NewLine    ; (dva 8b soucty bez prenosu - CHYBA)

    mov ax,  1000             ; AX = 00000011 11101000 =  1000
    mov bx, -2000             ; BX = 11111000 00110000 = -2000
    add al, bl                ; AX = 00000011 00011000 =   792
    adc ah, bh                ; AX = 11111100 00011000 = -1000
    call WriteInt16NewLine    ; (dva 8b soucty, prvni bez prenosu)
    ret                       ; (a druhy s prenosem - spravne)
```

Na vyzkoušení (2/2)

64b součet implementovaný pomocí dvou 32b součtů:

```
%include 'rw32.inc'          ;knihovna pro vstup a vystup
section .bss                 ;ne-inicializovany segment
    Z resq 1                 ; jedna 64 promenna Z (bez hodnoty)

section .data                ;inicializovany segment
    X dq  1000000000000      ; 64b promenna X s hodnotou  1000000000000
    Y dq -2000000000000      ; 64b promenna Y s hodnotou -2000000000000

section .text                ;kodovy segment
main:
    mov   eax, [X+0]         ; EAX = spodni polovina 64b promenne X
    mov   ebx, [Y+0]         ; EBX = spodni polovina 64b promenne Y
    add   eax, ebx           ; prvni soucet (bez prenosu)
    mov [Z+0], eax           ; spodni polovina promenne Z = EAX
                             ;  Z =      727379968
    mov   eax, [X+4]         ; EAX = horni  polovina 64b promenne X
    mov   ebx, [Y+4]         ; EBX = horni  polovina 64b promenne Y
    adc   eax, ebx           ; druhy soucet (s prenosem)
    mov [Z+4], eax           ; horni  polovina promenne Z = EAX
    ret                      ;  Z = -1000000000000
```

Posuvy a rotace

Základní posuvy a rotace

Posuvy (SHL, SHR) a rotace (ROL, ROR) umožňují pohybovat s bity uvnitř registru:

- První operand říká s čím hýbeme – libovolný registr nebo proměnná.
- Druhý operand říká o kolik bitů – konstanta nebo registr CL.
- Směr pohybu je součástí názvu instrukce.

Při posuvu se do registru doplňují nuly, při rotaci se doplňuje odcházející bit:

SHL (SHift Left)

```
mov  al,  60 ; AL = 00111100
shl  al,   1 ; AL = 01111000
shl  al,   1 ; AL = 11110000
shl  al,   1 ; AL = 11100000
shl  al,   1 ; AL = 11000000
```

SHR (SHift Right)

```
mov  bl,  60 ; BL = 00111100
shr  bl,   1 ; BL = 00011110
shr  bl,   1 ; BL = 00001111
shr  bl,   1 ; BL = 00000111
shr  bl,   1 ; BL = 00000011
```

ROL (ROtate Left)

```
mov  cl,  60 ; CL = 00111100
rol  cl,   1 ; CL = 01111000
rol  cl,   1 ; CL = 11110000
rol  cl,   1 ; CL = 11100001
rol  cl,   1 ; CL = 11000011
```

ROR (ROtate Right)

```
mov  dl,  60 ; DL = 00111100
ror  dl,   1 ; DL = 00011110
ror  dl,   1 ; DL = 00001111
ror  dl,   1 ; DL = 10000111
ror  dl,   1 ; DL = 11000011
```

Rozšířené posuvy a rotace

Aritmetické posuvy (SAL, SAR) se podobají obyčejným posuvům (SHL, SHR):

- Při posuvu doprava se ale zleva nedoplňuje nula, ale opakuje se nejlevější bit.
- Aritmetický posuv o jednu pozici se podobá násobení nebo dělení dvěma.

SAL (Shift Arithmetic Left)

```
mov  al,  15 ;AL = 00001111 =  15
sal  al,   1 ;AL = 00011110 =  30
mov  al, -16 ;AL = 11110000 = -16
sal  al,   1 ;AL = 11100000 = -32
```

SAR (Shift Arithmetic Right)

```
mov  bl,  15 ;BL = 00001111 =  15
sar  bl,   1 ;BL = 00000111 =   7
mov  bl, -16 ;BL = 11110000 = -16
sar  bl,   1 ;BL = 11111000 =  -8
```

Rotace skrz přenos (RCL, RCR) se podobají obyčejným rotacím (ROL, ROR):

- Při rotaci se do registru nedoplňuje odešlý bit, ale předchozí hodnota příznaku CF.
- Registry tedy rotují jako by byly o jeden bit (příznak CF) delší.

RCL (Rotate through Carry Left)

```
mov  cl, 102 ;CL= 01100110, CF= ?
rcl  cl,   1 ;CL= 1100110?, CF= 0
rcl  cl,   1 ;CL= 100110?0, CF= 1
rcl  cl,   1 ;CL= 00110?01, CF= 1
```

RCR (Rotate through Carry Right)

```
mov  dl, 102 ;DL= 01100110, CF= ?
rcr  dl,   1 ;DL= ?0110011, CF= 0
rcr  dl,   1 ;DL= 0?011001, CF= 1
rcr  dl,   1 ;DL= 10?01100, CF= 1
```

Na vyzkoušení

Rotace umožňují vyměňovat jednotlivé části registru:

- Rotace o 4 bity je rotace o jednu číslici v šestnáctkové soustavě.
- Rotace N-bitového registru o N/2 bitů vymění jeho horní a spodní polovinu.
- Rotace N-bitového registru o N bitů obnoví jeho původní stav.

```
section .text          ;kodovy segment
main:
    mov  ax, 0xABCD ; 1010101111001101 = ABCD   celkem rotace  0 bitu
    ror  ax, 1      ; 1101010111100110 = D5E6   celkem rotace  1 bitu
    ror  ax, 1      ; 0110101011110011 = 6AF3   celkem rotace  2 bitu
    ror  ax, 1      ; 1011010101111001 = B579   celkem rotace  3 bitu
    ror  ax, 1      ; 1101101010111100 = DABC   celkem rotace  4 bitu
                    ; rotace o 4 bity je rotace o jednu 16-kovou cislici
    ror  ax, 1      ; 0110110101011110 = 6D5E   celkem rotace  5 bitu
    ror  ax, 1      ; 0011011010101111 = 36AF   celkem rotace  6 bitu
    ror  ax, 1      ; 1001101101010111 = 9B57   celkem rotace  7 bitu
    ror  ax, 1      ; 1100110110101011 = CDAB   celkem rotace  8 bitu
                    ; 16b registr zrotovany o 8 bitu je xchg AH, AL
    ror  ax, 4      ; 1011110011011010 = BCDA   celkem rotace 12 bitu
    ror  ax, 4      ; 1010101111001101 = ABCD   celkem rotace 16 bitu
    ret             ; 16b registr zrotovany o 16 bitu je puvodni stav
```

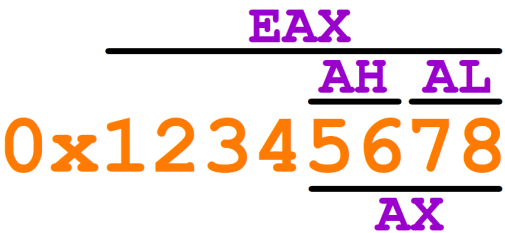
Na procvičení

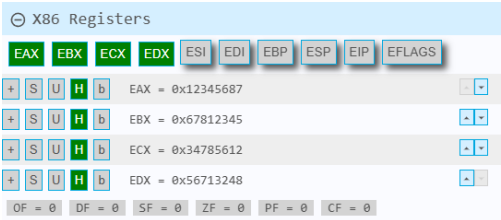
Vyzkoušejte si:

- Registry EAX, EBX, ECX a EDX si inicializujte na hodnoty:

```
mov eax, 0x12345687
mov ebx, 0x67812345
mov ecx, 0x34785612
mov edx, 0x56713248
```

- Pomocí rotací obsah všech registrů změňte na hodnotu 0x12345678.





Nápověda:

- Velikost rotací se udává v počtu bitů, jedna číslice v 16-kové soustavě jsou 4 bity.
- Debugger obsah registrů umí zobrazit i jako čísla v šestnáctkové soustavě.

Násobení a dělení

Problémy s násobením a dělením

Při násobení a dělení čísel dochází ke změně jejich velikosti (datového typu):

- Na rozdíl od sčítání a odčítání, výsledek násobení a dělení vyjde jinak, podle toho jestli čísla začínající jedničkou interpretujeme jako záporná nebo velká kladná.

U obou operací uvádíme jen jeden (zdrojový) operand:

- Cílový operand je určen implicitně – podle velikosti použité instrukce.
- To jestli počítáme se znaménkem nebo bez znaménka je součástí názvu instrukce.

MUL (MULtiply) bez znaménka
IMUL (Integer MULtiply) se znaménkem

```
mul   bl  ;      AX  =  AL * BL
mul   bx  ;   DX:AX  =  AX * BX
mul  ebx  ;  EDX:EAX = EAX * EBX
```

DIV (DIVide) bez znaménka
IDIV (Integer DIVide) se znaménkem

```
div   bl  ;  AL =      AX  / BL
div   bx  ;  AX =   DX:AX  / BX
div  ebx  ; EAX = EDX:EAX  / EBX
```

Dvojtečka znamená že velké (32b, 64b) číslo je rozděleno do dvou menších registrů:

- DX:AX – horních 16b je v DX, spodních 16b je v AX.
- EDX:EAX – horních 32b je v EDX, spodních 32b je v EAX.

Násobení

Procesor umí násobit se znaménkem (IMUL) a bez-znaménka (MUL):

- Při násobení dochází ke zvětšení datového typu (na dvojnásobek).
- Násobení má pouze jeden operand, číslo kterým násobíme.
- Umístění čísla které násobíme a výsledku jsou dány velikostí instrukce.

```
imul  bl        ;      AX  =  AL * BL   ( 8b nasobeni, se znamenkem)
imul  bx        ;   DX:AX  =  AX * BX   (16b nasobeni, se znamenkem)
imul ebx        ;  EDX:EAX = EAX * EBX  (32b nasobeni, se znamenkem)
```

Pozor, výsledek 16b násobení je rozdělen mezi dva registry (DX:AX):

- Horní polovina bude výsledku se uloží do DX, spodní polovina do AX.
- Poloviny výsledku můžeme spojit pomocí posuvu nebo rotace.

Procesor umí násobit registrem nebo proměnnou, ale ne konstantou:

```
mul bl          ;ok     - AX = AL * BL   (8b nasobeni, bez znamenka)
mul byte [X]    ;ok     - AX = AL * [X]  (8b nasobeni, bez znamenka)
mul [X]         ;CHYBA - nelze urcit velikost instrukce
mul byte 10     ;CHYBA - procesor neumi nasobit konstantou
```

Na vyzkoušení

Ukázka 8b násobení se znaménkem a bez znaménka:

```
%include 'rw32.inc'          ;knihovna pro vstup a vystup

section .text                ;kodovy segment
main:
                             ; nasobeni se znamenkem
    mov  al, 0b11111111      ; AL = 11111111          =   -1 =   255
    mov  bl, 10              ; BL = 00001010          =   10 =    10
    imul bl                  ; AX = AL * BL
    call WriteInt16NewLine   ; AX = 11111111 11110110 =  -10 = 65526

                             ; nasobeni bez znamenka
    mov  al, 0b11111111      ; AL = 11111111          =   -1 =   255
    mov  bl, 10              ; BL = 00001010          =   10 =    10
    mul  bl                  ; AX = AL * BL
    call WriteUInt16NewLine  ; AX = 00001001 11110110 = 2555 =  2555

    ret
```

Spojení výsledku 16b násobení

```
;prvni 16b cinitel je v AX
;druhy 16b cinitel je v BX


;registr AX vynasob registrem BX
;32b vysledek nasobeni je v DX:AX
    mul   bx


;registr EAX rotuj o 16b (doleva)
;(vymeni horni a spodni polovinu)
    rol   eax,  16


;hornich 16b zkopiruj z DX do AX
    mov   ax,  dx


;registr EAX rotuj o 16b (doleva)
;(vymeni horni a spodni polovinu)
    rol   eax,  16
```

EAX	????????????????	16b CINITEL
EDX	????????????????	????????????????

EAX	????????????????	SPODNICH 16b
EDX	????????????????	HORNICH 16b

EAX	SPODNICH 16b	????????????????
EDX	????????????????	HORNICH 16b

EAX	SPODNICH 16b	HORNICH 16b
EDX	????????????????	HORNICH 16b

EAX	HORNICH 16b	SPODNICH 16b
EDX	????????????????	HORNICH 16b

Na vyzkoušení

Ukázka 16b násobení se znaménkem :

```
%include 'rw32.inc'          ;knihovna pro vstup a vystup
section .data                ;inicializovany segment
    X dw  1000               ; 16b promenna X s hodnotou  1000
    Y dw -1000               ; 16b promenna Y s hodnotou -1000
section .text                ;kodovy segment
main:
    mov   ax, [X]            ; AX = [X]
    mov   bx, [Y]            ; BX = [Y]
    imul  bx                 ; DX:AX = AX * BX   (se znamenkem)
                             ; EAX = ???????? ???????? 10111101 11000000
                             ; EDX = ???????? ???????? 11111111 11110000
    call WriteInt32NewLine   ; CHYBA - funkce vypisuje 32b hodnotu z EAX
                             ;   ale vysledek nasobeni je v DX:AX

    rol  eax,  16            ; rotaci vymen horni a spodni polovinu EAX
    mov   ax,  dx            ; hornich 16b vysledku zkopiruj z DX do AX
    rol  eax,  16            ; rotaci vymen horni a spodni polovinu EAX
                             ; EAX = 11111111 11110000 10111101 11000000
    call WriteInt32NewLine   ; vypis hodnotu z EAX (vysledek nasobeni)
    ret
```

Na procvičení

Vyzkoušejte si:

- Vytvořte si tři 8b proměnné (R, S, T), 16b proměnnou (U) a 32b proměnnou (V).
- Bez-znaménkově spočítejte následující rovnice, a výsledek uložený v proměnných U a V si zobrazte v debuggeru.

$$U = S * T \qquad V = R * S * T$$

Například:

$$R = 10 \qquad R = 150$$
$$S = 20 \qquad S = 200$$
$$T = 30 \qquad T = 250$$

$$U = 600 \qquad U = 50\,000$$
$$V = 6\,000 \qquad V = 7\,500\,000$$

Nápověda:

- Při násobení oba činitelé musejí mít stejnou velikost (může být potřeba rozšíření).
- Výsledek 16b (ale ne 8b) násobku je rozpůlen do dvou 16b registry (DX:AX), které je před uložením do paměti potřeba spojit do jednoho 32b čísla.
- Nezapomeňte si v debuggeru nastavit zobrazování čísel bez-znaménka.

Dělení

Procesor umí celočíselně dělit se znaménkem (IDIV) a bez-znaménka (DIV):

- Při dělení dochází ke zmenšení datového typu (na polovinu).
- Dělení má pouze jeden operand, číslo kterým dělíme.
- Umístění čísla které dělíme, podílu a zbytku jsou dány velikostí instrukce.

```
idiv  bl      ;  AL =     AX  / BL  (podil),  AH =     AX  % BL  (zbytek)
idiv  bx      ;  AX =  DX:AX  / BX  (podil),  DX =  DX:AX  % BX  (zbytek)
idiv ebx      ; EAX = EDX:EAX / EBX (podil), EDX = EDX:EAX % EBX (zbytek)
```

Pozor, dělenec 16b dělení je umístěn ve dvou registrech (DX:AX):

- Horní polovina dělence musí být v DX, spodní polovina v AX.
- Po dělení bude horní polovině zbytek a ve spodní polovině podíl.

Pokud se podíl nevejde do registrů, nebo zkusíme dělit nulou, program **HAVARUJE**:

- Podobně jako u násobení, procesor také neumí dělit konstantou.

```
div byte 10  ; CHYBA - procesor neumi delit konstantou
mov  bl, 0   ; BL = 0
div bl       ; CHYBA - deleni nulou havaruje
```

Na vyzkoušení

Ukázka dělení se znaménkem a bez znaménka (které havaruje):

```
%include 'rw32.inc'                ;knihovna pro vstup a vystup

section .text                      ;kodovy segment
main:
                                   ; deleni se znamenkem
    mov  ax, 0b1111111111110110    ; Ax = 11111111 11110110 = -10 = 65526
    mov  bl, 0b00000010            ; BL = 00000010          =   2 =     2
    idiv bl                        ; AH = AX % BL,   AL = AX / BL
    call WriteBin8NewLine          ; AL = 11111011          =  -5 =   251
                                   ; AH = 00000000          =   0 =     0

                                   ; deleni bez znamenka
    mov  ax, 0b1111111111110110    ; Ax = 11111111 11110110 = -10 = 65526
    mov  bl, 0b00000010            ; BL = 00000010          =   2 =     2
    div  bl                        ; AH = AX % BL,   AL = AX / BL (HAVARIE)
    call WriteBin8NewLine          ;  vysledek 65526 / 2 = 32763
                                   ;  se nevejde do 8b registru AL
    ret
```

Rozpůlení dělence 16b dělení

```
;32b delenec je v EAX
;16b delitel je v  BX


;registr EAX rotuj o 16b (doleva)
;(vymeni horni a spodni polovinu)
    rol   eax,  16


;hornich 16b zkopiruj z AX do DX
    mov    dx,  ax


;registr EAX rotuj o 16b (doleva)
;(vymeni horni a spodni polovinu)
    rol   eax,  16


;registry DX:AX vydel pomoci BX
;podil je v AX, zbytek je v DX
    div    bx
```

EAX	HORNICH 16b	SPODNICH 16b
EDX	????????????????	????????????????

EAX	SPODNICH 16b	HORNICH 16b
EDX	????????????????	????????????????

EAX	SPODNICH 16b	HORNICH 16b
EDX	????????????????	HORNICH 16b

EAX	HORNICH 16b	SPODNICH 16b
EDX	????????????????	HORNICH 16b

EAX	HORNICH 16b	16b PODIL
EDX	????????????????	16b ZBYTEK

Na procvičení

Vyzkoušejte si:

- Vytvořte si tři 8b proměnné (K, L, M) a dvě 16b proměnné (V, Z).
- Znaménkově spočítejte výsledek následující rovnice. Podíl uložte do proměnné V, zbytek do proměnné Z, a obsah obou proměnných si zkontrolujte v debuggeru.

$$\frac{K * -40}{L * M}$$

Například:

$$K = -1 \qquad L = 3 \qquad M = 5$$

$$V = 2 \qquad Z = 10$$

$$K = 100 \qquad L = 12 \qquad M = -5$$

$$V = 66 \qquad Z = -40$$

Nápověda:

- Procesor neumí násobit ani dělit konstantami.
- Při 16b dělení má dělenec velikost 32b a musí rozpůlen do registrů DX:AX.
- Pří počítání se znaménkem můžeme pro rozšíření používat instrukce CBW, CWD, CDQ a CWDE (při počítání bez znaménka rozšiřujeme nulami).

Na procvičení (na doma)

Vyzkoušejte si:

- Vytvořte si pole 8b proměných (X = 100, 110, 120, -100) a 64b proměnnou (Y).
- Znaménkově spočítejte násobek všech čtyř položek pole X, a výsledek uložte do Y.
- Výslednou hodnotu proměnné Y si zobrazte v debuggeru.

Například:

$$[X+0] = 100$$
$$[X+1] = 110$$
$$[X+2] = 120$$
$$[X+3] = -100$$

$$Y = -132\,000\,000$$

Nápověda:

- Násobené hodnoty může být potřeba rozšířit na správnou velikost.
- Spodní polovinu 64b výsledku je potřeba uložit na adresu Y+0, horní na Y+4.