Procesor, vývojové nástroje a základy programování

ISU-cv02

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

21. února 2025

Procesor

Procesor

Procesor (CPU) je centrální součástka počítače:

- Výpočty provádí jeho aritmeticko-logická jednotka (ALU) ovládaná řadičem (CCU).
- Procesor pracuje pouze s celými čísly (bez znaménka nebo v doplňkovém kódu).
- Pro práci s desetinnými čísly je zapotřebí matematický koprocesor (FPU, viz. cv11).

Procesor může pracovat buď v 16b (základním) nebo 32b (chráněném) režimu:

- Na cvičeních vždy pracujeme s procesory z rodiny x86 v 32b režimu.
- Výpočet fyzické adresy v základním 16b režimu (p3s14–17) bývá u zkoušky.
- Na hypotetické počítače z druhé přednášky (p2s7–34) můžete zapomenout.

Program procesoru píšeme v jazyce symbolických adres, lidově zvaném assembler:

- Hodnoty neukládáme do proměnných, ale do registrů.
- Hodnoty nezpracováváme pomocí výrazů a příkazů, ale instrukcí.

Registry

Registr je malá, rychlá paměťová jednotka umístěná uvnitř těla procesoru:

- Procesor obsahuje celkem deset 32b registrů které nás zajímají.
- Každý registr má nějaké jméno, velikost a konkrétní účel.
  Například, pro vstup a výstup vždy používáme EAX, pro cyklus vždy ECX.
- Názvy registrů jsou case-insensitive – můžeme je psát malými i velkými písmeny.

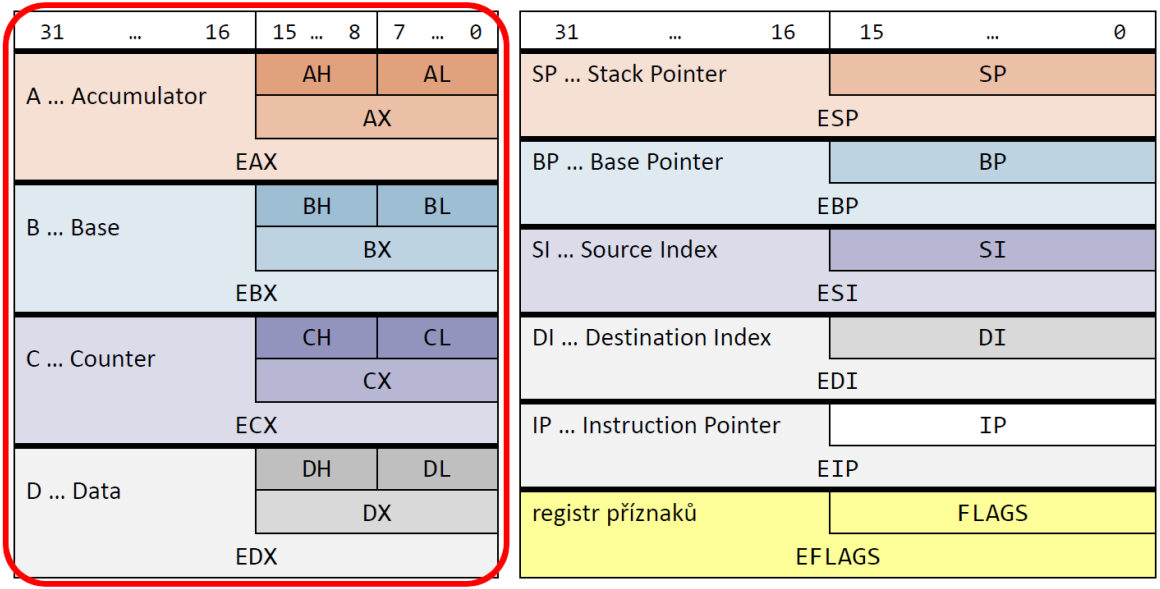
Pro aritmeticko-logické operace jsou nám k dispozici čtyři obecné 32b registry:

- Accumulator (EAX), Base (EBX), Counter (ECX) a Data (EDX).
- Pokud chceme pracovat s menšími něž 32b čísly, místo celých registrů můžeme používat jejich spodní polovinu (16b) nebo dvě spodní čtvrtiny (8b).
- Spodní poloviny registrů (16b) se nazývají AX, BX, CX a DX.
- Spodní čtvrtiny registrů (8b) se nazývají AH, BH, CH, DH a AL, BL, CL, DL.

Pozor EAX, AX, AH, AL **NEJSOU** čtyři nezávislé registry, ale čtyři různé části toho samého registru, které se spolu vzájemně překrývají:

- Například, pokud změníme hodnotu uloženou v registru AL, změní se tím i hodnota kterou uvidíme v registrech AX a EAX (hodnota v AH se nezmění).

Registry procesoru - obecné registry



Na ukázku

Sledujte obsah uvedených registrů ve dvojkové a desítkové soustavě se znaménkem:

- Hodnotu registru můžeme změnit instrukcí MOV (přiřazení, =).
- Počáteční hodnota všech registrů je nedefinovaná.

```
                ;|-----------------|
                ;|     AX =   0    |
mov   ax,    0  ;|00000000|00000000|
                ;|AH =   0|AL =   0|
                ;|-----------------|
                ;|     AX =   1    |
mov   al,    1  ;|00000000|00000001|
                ;|AH =   0|AL =   1|
                ;|-----------------|
                ;|     AX =   0    |
mov   ax,    0  ;|00000000|00000000|
                ;|AH =   0|AL =   0|
                ;|-----------------|
                ;|     AX = 256    |
mov   ah,    1  ;|00000001|00000000|
                ;|AH =   1|AL =   0|
                ;|-----------------|
```

```
                ;|-----------------|
                ;|     BX = ????   |
mov   bl,    0  ;|????????|00000000|
                ;|BH = ???|BL =   0|
                ;|-----------------|
                ;|     BX = ????   |
mov   bl,  0xF  ;|????????|00001111|
                ;|BH = ???|BL =  15|
                ;|-----------------|
                ;|     BX = 3855   |
mov   bh,   bl  ;|00001111|00001111|
                ;|BH =  15|BL =  15|
                ;|-----------------|
                ;|     BX =   -1   |
mov   bx,   -1  ;|11111111|11111111|
                ;|BH =  -1|BL =  -1|
                ;|-----------------|
```

Na procvičení

Určete obsah uvedených registrů ve dvojkové a desítkové soustavě se znaménkem:

```
                ;|-----------------|
                ;|      CX = ????  |
mov   cx, 128   ;|????????|????????|
                ;|CH = ???|CL = ???|
                ;|-----------------|
                ;|      CX = ????  |
mov   cl,  -1   ;|????????|????????|
                ;|CH = ???|CL = ???|
                ;|-----------------|
                ;|      CX = ????  |
mov   cx, 256   ;|????????|????????|
                ;|CH = ???|CL = ???|
                ;|-----------------|
                ;|      CX = ????  |
mov   cl,  -1   ;|????????|????????|
                ;|CH = ???|CL = ???|
                ;|-----------------|
```

```
                ;|-----------------|
                ;|      DX = ????  |
mov   dl,0b10   ;|????????|????????|
                ;|DH = ???|DL = ???|
                ;|-----------------|
                ;|      DX = ????  |
mov   dl,101o   ;|????????|????????|
                ;|DH = ???|DL = ???|
                ;|-----------------|
                ;|      DX = ????  |
mov   dx,0xFE   ;|????????|????????|
                ;|DH = ???|DL = ???|
                ;|-----------------|
                ;|      DX = ????  |
mov   dl, 256   ;|????????|????????|
                ;|DH = ???|DL = ???|
                ;|-----------------|
```

Instrukce

Instrukce jsou elementární příkazy jazyka procesoru:

- Každá instrukce má nějaké jméno, a některé z nich mohou mít také operandy.
- Názvy instrukcí jsou case-insensitive – můžeme je psát malými i velkými písmeny.
- Operandy instrukcí se oddělují čárkou, a na jejich pořadí záleží.
  První operand je obvykle cíl (destination), druhý operand je obvykle zdroj (source).

Seznam dostupných instrukcí je definován instrukční sadou procesoru:

- Základní instrukce jsou ke stažení v tabulce, kompletní seznam je v manuálu.

```
mov  eax,  10    ;prirazeni     EAX = 10
add  eax,  20    ;soucet        EAX = EAX + 20
sub  eax,  30    ;rozdil        EAX = EAX - 30
neg  eax         ;negace        EAX = -EAX      (dvojkovy doplnek)
xchg eax, ebx    ;vymena        swap(EAX,EBX)
```

Většina instrukcí existuje ve třech různých velikostech (8b, 16b, 32b):

- To kterou velikost použít automaticky odvozuje překladač – podle operandů.
- Operandy instrukce proto obvykle musejí mít stejnou velikost (viz. cv04).

```
mov   ax,  al    ;CHYBA -- registr AX je 16b, registr AL je 8b
```

Na procvičení

Určete obsah uvedených registrů ve dvojkové a desítkové soustavě se znaménkem:

```
                ;|      AX = ????    |
mov   ax,0xFF   ;|????????|????????|
                ;|AH = ???|AL = ???|
                ;|-----------------|
                ;|      AX = ????    |
add   ax,  ax   ;|????????|????????|
                ;|AH = ???|AL = ???|
                ;|-----------------|
                ;|      BX = ????    |
mov   bx, 256   ;|????????|????????|
                ;|BH = ???|BL = ???|
                ;|-----------------|
                ;|      BX = ????    |
add   bl, 128   ;|????????|????????|
                ;|BH = ???|BL = ???|
                ;|-----------------|
                ;|      BX = ????    |
add   bl, 128   ;|????????|????????|
                ;|BH = ???|BL = ???|
```

```
                ;|      CX = ????    |
mov   cx,-255   ;|????????|????????|
                ;|CH = ???|CL = ???|
                ;|-----------------|
                ;|      CX = ????    |
xchg ch,   cl   ;|????????|????????|
                ;|CH = ???|CL = ???|
                ;|-----------------|
                ;|      DX = ????    |
mov   dx,    1  ;|????????|????????|
                ;|DH = ???|DL = ???|
                ;|-----------------|
                ;|      DX = ????    |
neg   dx        ;|????????|????????|
                ;|DH = ???|DL = ???|
                ;|-----------------|
                ;|      DX = ????    |
sub   dh, 128   ;|????????|????????|
                ;|DH = ???|DL = ???|
```

Vývojové nástroje a základy programování

Vývojové nástroje

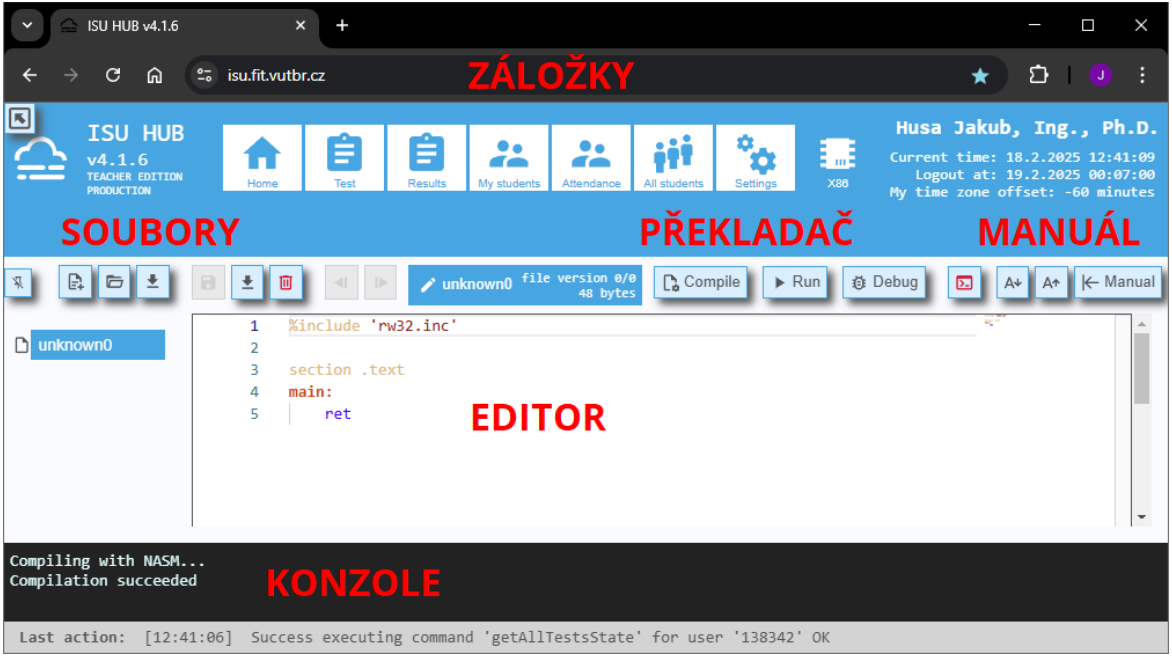
K programování na strojové úrovni budeme potřebovat:

- Editor ve kterém budeme psát program.
- Překladač kterým program přeložíme do strojového kódu.
- Linker kterým strojový kód spojíme s knihovnami na spustitelný soubor.
- Debugger kterým program můžeme krokovat abychom v něm odhalili chyby.

Situaci si výrazně usnadníme použitím nějakého vývojového prostředí:

- Osobně vám doporučuji používat simulátor x86 v ISU-HUBu, ve kterém také budete psát vaše bodované testy (viz. cv01).
- Pro off-line programování doporučuji stáhnout a používat SASM (není potřeba).
- Další možnosti najdete v E-learningu v osnově 2. cvičení.

Vývojové prostředí ISU-HUB



Struktura programu

Paměťový prostor programu se dělí na pět segmentů:

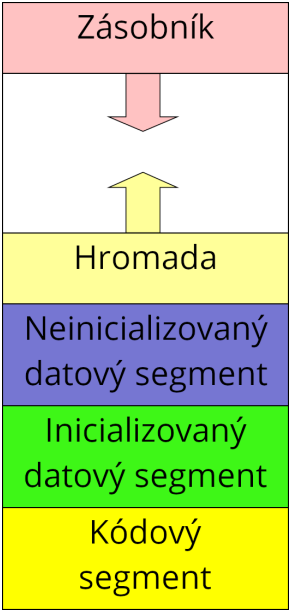
- Instrukce píšeme do funkcí, které píšeme do kódového segmentu označovaného jako section .text.
- Kódový segment musí obsahovat hlavní funkci (main) pojmenovanou main:, _main:, nebo CMAIN:, v závislosti na zvoleném vývojovém prostředí.

Funkce začínají návěštím a končí instrukcí RET (return):

- Návratem z funkce main se ukončí celý program.
- Komentáře se označují středníkem (;).

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

section .text       ;kodovy segment
main:               ;zacatek funkce main
                    ;telo    funkce main
    ret             ;konec   funkce main
```



Vstup a výstup programu

Nad kódovým segmentem mohou být umístěny datové segmenty (viz. cv03) a nad nimi direktiva %include, která umožňuje do programu vkládat knihovny:

- Funkce pro vstup a výstup programu se nacházejí v knihovně rw32.inc.
- Seznam dostupných funkcí najdete v E-learningu v osnově 7. cvičení.

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

Funkce voláme instrukcí CALL:

- Názvy funkcí jsou case-sensitive – na psaní malých a velkých písmen záleží!
- Knihovní funkce čtou a vypisují pouze z registrů AL (8b), AX (16b), nebo EAX (32b).
- Pro přesun (přiřazení) hodnot mezi registry používáme instrukci MOV.

```
call ReadInt32            ;ze vstupu nacti 32b do EAX
call WriteInt32           ;na vystup vypis 32b  z EAX
call WriteNewLine         ;na vystup vypis konec radku
call ReadInt32NewLine     ;ze vstupu nacti 32b do EAX a vypis konec radku
call WriteInt32NewLine    ;na vystup vypis 32b  z EAX a vypis konec radku
```

- Popis všech instrukcí a funkcí na ISU-HUBu otevřeme tlačítkem: Manual

Na vyzkoušení

Program ze vstupu načte dvě 32b čísla (X a Y) a vypíše jejich součet (X+Y):

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

section .text                ;kodovy segment
main:                        ; zacatek funkce main (zacatek programu)

                             ;                     | EAX EBX ECX EDX |
    call ReadInt32NewLine    ; nacti 32b do EAX    |  X  ??? ??? ??? |
    mov  ebx, eax            ; EBX = EAX           |  X   X  ??? ??? |
    call ReadInt32NewLine    ; nacti 32b do EAX    |  Y   X  ??? ??? |
    add  eax, ebx            ; EAX = EAX + EBX     | Y+X  X  ??? ??? |
    call WriteInt32NewLine   ; vypis hodnotu EAX   | Y+X  X  ??? ??? |

    ret                      ; konec funkce main (konec programu)
```

Na procvičení

Vyzkoušejte si:

- Ze vstupu si načtěte dvě 32b čísla (X a Y).
- Spočítejte hodnoty X-Y, -X+Y, -X-Y, a vypište je na výstup.

Například:

- 10, 20 => -10
  10
  -30
- 10, -10 => 20
  -20
  0

Nápověda:

- Hodnoty X a Y si po načtení můžeme zkopírovat i do registrů ECX a EDX.
- Hodnotu registru EAX si po výpočtu můžeme vynulovat přiřazením konstanty 0.

```
mov   ecx, eax    ;prirazeni    ECX = EAX
mov   eax,   0    ;prirazeni    EAX =   0
```

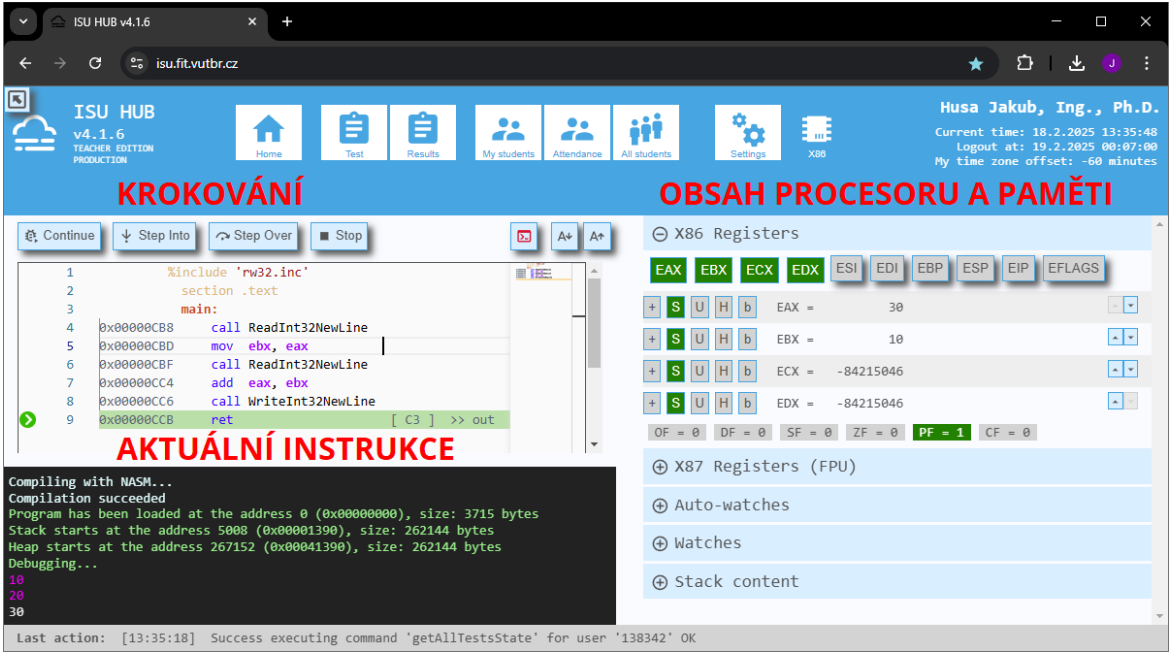
Ladění programu - ISU-HUB

Pro ladění programu je nám k dispozici debugger:

- Vlevo je zvýrazněná aktuální instrukce, která se provede v následujícím kroku.
- Vpravo se zobrazuje obsah všech registrů CPU, proměnných (viz. cv03), příznaků (viz. cv05), zásobníku (viz. cv08), a registrů FPU (viz. cv11).
- Podobně jako u vyšších programovacích jazyků, program můžeme krokovat po jednotlivých instrukcích, a nebo si v něm můžeme nastavit breakpoint.

Registry si můžeme vypsat znaménkově, bez-znaménkově, šestnáctkově a binárně:

Debugger - ISU-HUB



Na procvičení

Vyzkoušejte si:

- Ze vstupu si načtěte čtyři 8b čísla (K, L, M, N).
- Spočítejte hodnoty K+L+M+N, 16K, 100-(M+N), a vypište je na výstup.

Například:

- 1, 2, 3, 4 => 10
16
93
- 10, 20, 30, 40 => 100
-96
30

Nápověda:

- Pro práci s 8b hodnotami používáme registry AH, AL, BH, BL, CH, CL, DH, DL.
- Násobení můžeme implementovat pomocí opakovaného sčítání.
- Pro načítání a vypisování 8b čísel nepoužíváme funkce Int32, ale Int8.

```
call ReadInt8NewLine      ;ze vstupu nacti  8b do  AL a vypis konec radku
call WriteInt8NewLine     ;na vystup vypis  8b  z  AL a vypis konec radku
```