Procesor, vývojové nástroje a základy programování

ISU-cv02

Ing. Jakub Husa

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

ihusa@fit.vutbr.cz

16. února 2023

Procesor

Procesor

Procesor (CPU) je centrální součástka počítače:

- Výpočty provádí jeho aritmeticko-logická jednotka (ALU) ovládaná řadičem.
- Procesor vždy pracuje s celými čísly v doplňkovém kódu.
- Práci s desetinnými čísly má na starosti matematický koprocesor (FPU, viz. cv11).

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

- Na cvičeních budeme vždy pracovat pouze 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:

- Hodnoty neukládáme do proměnných, ale do registrů.
- Hodnoty nezpracováváme pomocí výrazů, 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 a konkrétní účel.
- Názvy registrů jsou case-insensitive, můžeme je psát malými i velkými písmeny.

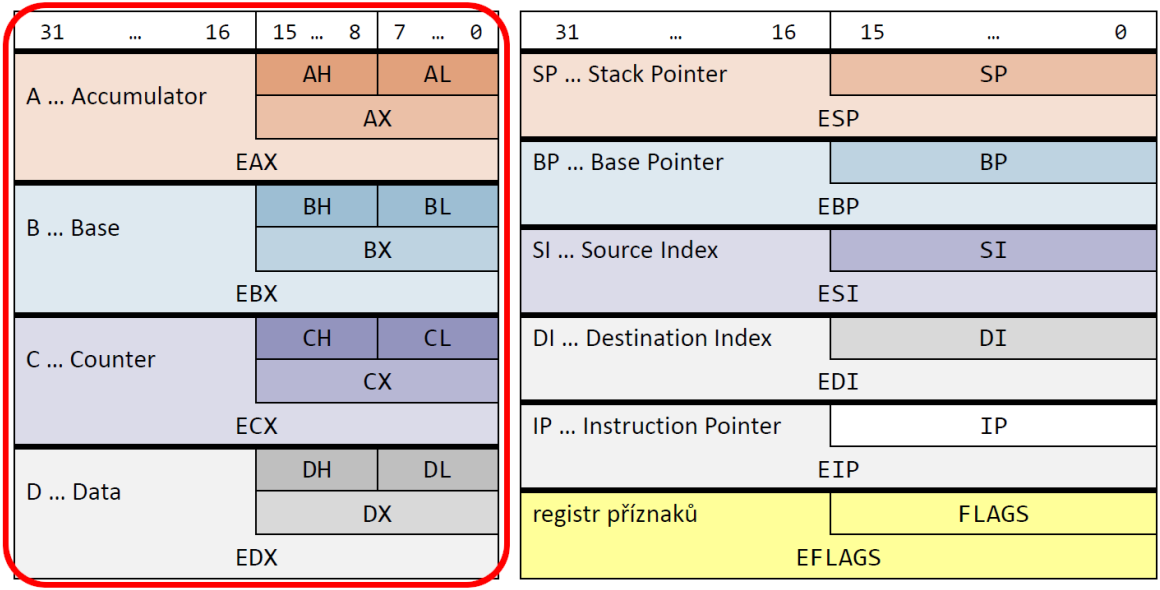
Pro matematické výpočty slouží čtyři obecné 32b registry:

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

Pozor AX, AH, AL, atd. **NEJSOU** další nezávislé registry:

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

Registry procesoru - obecné registry



Jak se bude měnit obsah uvedených registrů v desítkové soustavě?

```
                ;|      AX =    0    |
mov   ax,    0  ;|00000000|00000000|
                ;|AH =    0|AL =    0|
                ;|-----------------|
                ;|      AX =    1    |
mov   ax,    1  ;|00000000|00000001|
                ;|AH =    0|AL =    1|
                ;|-----------------|
                ;|      AX =    2    |
mov   ax,    2  ;|00000000|00000010|
                ;|AH =    0|AL =    2|
                ;|-----------------|
                ;|      AX =    0    |
mov   al,    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|
                ;|-----------------|
                ;|      BX = -256    |
mov   bl,    0  ;|11111111|00000000|
                ;|BH =   -1|BL =    0|
```

Instrukce jsou elementární příkazy určující činnost procesoru:

- Každá instrukce má nějaké jméno, a může mít i nějaké operandy.
- Operandy instrukce oddělujeme čárkou a na jejich pořadí záleží.
- Některé instrukce umějí pracovat jen s konkrétními registry (násobení, viz. cv04).
- Většina instrukcí existuje ve třech variantách (8b, 16b, 32b), kterou z nich použít většinou automaticky odvozuje překladač podle velikosti použitých operandů.

Dostupné instrukce jsou definovány instrukční sadou procesoru:

```
1 mov  eax,  10   ;prirazeni     EAX = 10
2 add  eax,  20   ;soucet        EAX = EAX + 20
3 sub  eax,  30   ;rozdil        EAX = EAX - 30
4 inc  eax        ;inkrementace  EAX = EAX + 1
5 dec  eax        ;dekrementace  EAX = EAX - 1
6 neg  eax        ;negace        EAX = -EAX     (dvojkovy doplnek)
```

- V E-learningu najdete seznam základních instrukcí které v ISU budeme používat.
- Pokud vás zajímá jak vypadá kompletní instrukční sada stáhněte si manuál.

Počáteční hodnota registrů je nedefinovaná:

```
1 add  eax,  10   ;EAX = ???
```

Hodnoty v jiné než desítkové soustavě označujeme předponou nebo příponou:

```
2 mov  eax, 0xF   ;EAX =  15
3 mov  eax, 10b   ;EAX =   2
```

Instrukce jsou vždy vykonávány v sekvenčním pořadí:

```
4 mov  eax,  10   ;EAX =  10
5 mov  ebx,  20   ;EAX =  10, EBX =  20
6 add  eax, ebx   ;EAX =  30, EBX =  20
```

Operandy vždy musejí mít stejnou velikost:

```
7 mov   ah,  al   ;ok     (AH = AL)
8 mov   ah,  ax   ;CHYBA (registr AH je 8b, registr AX je 16b)
```

Vyzkoušejte si:

- Jak se bude měnit obsah uvedených registrů v desítkové soustavě?

```
1  mov   ax,   0    ; AH =   0, AL =   0, AX =   0
2  inc   ah         ; AH = ???, AL = ???, AX = ???
3  add   ax, 128    ; AH = ???, AL = ???, AX = ???
4  add   al, 128    ; AH = ???, AL = ???, AX = ???
5
6  mov   bx,   0    ; BH =   0, BL =   0, BX =   0
7  add   bl,0x0F    ; BH = ???, BL = ???, BX = ???
8  add   bl,0xF0    ; BH = ???, BL = ???, BX = ???
9  add   bx,  bx    ; BH = ???, BL = ???, BX = ???
10
11 mov   cx,   0    ; CH =   0, CL =   0, CX =   0
12 dec   ch         ; CH = ???, CL = ???, CX = ???
13 sub   cl, 128    ; CH = ???, CL = ???, CX = ???
14 neg   cx         ; CH = ???, CL = ???, CX = ???
15
16 mov   dx,   0    ; DH =   0, DL =   0, DX =   0
17 dec   dx         ; DH = ???, DL = ???, DX = ???
18 neg   dh         ; DH = ???, DL = ???, DX = ???
19 add   dl,  dh    ; DH = ???, DL = ???, DX = ???
```

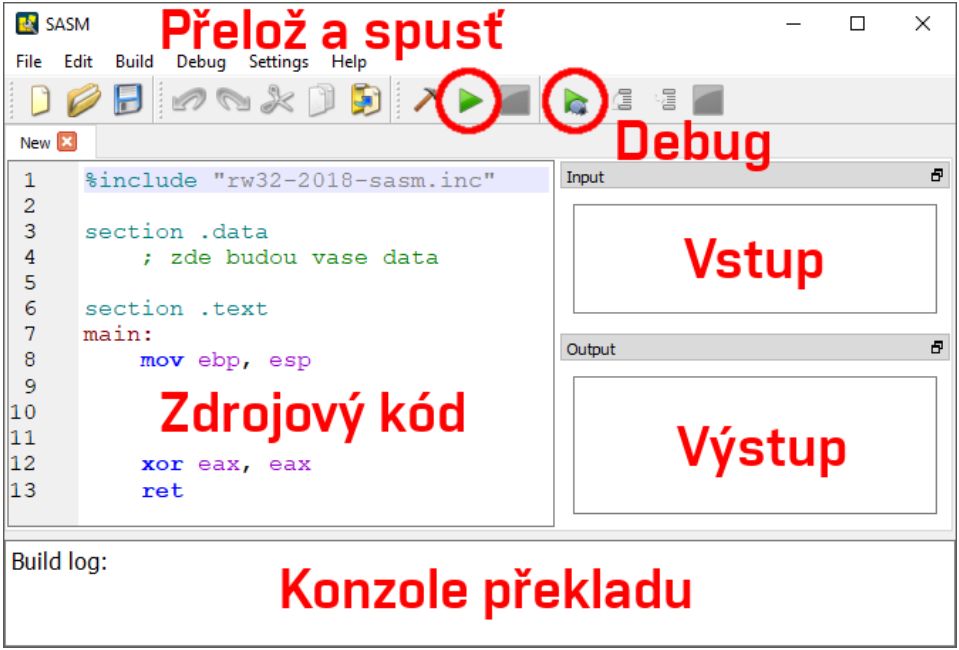
Vývojové nástroje a základy programování

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 si z E-learningu stáhnout SASM pro Windows.
- Používat můžete také simulátor x86 v ISU-HUB.
- Další možnosti najdete v osnově 2. cvičení.


SASM
Přelož a spusť
File Edit Build Debug Settings Help
Debug
New
1 %include "rw32-2018-sasm.inc"
2
3 section .data
4 ; zde budou vase data
5
6 section .text
7 main:
8 mov ebp, esp
9
10
11
12 xor eax, eax
13 ret
Zdrojový kód
Input
Vstup
Output
Výstup
Build log:
Konzole překladu

Paměťový prostor programu se dělí na pět segmentů:

- Instrukce zapisujeme do kódového segmentu označeného jako "section .text"
- Kódový segment musí obsahovat hlavní funkci programu označenou návěštím "main:" nebo "_main:" nebo "CMAIN:" (záleží na zvoleném vývojovém prostředí).

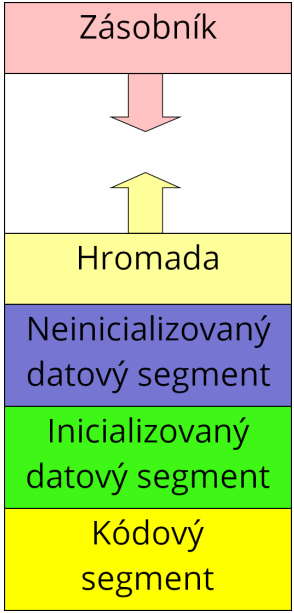
Funkce začínáme vytvořením zásobníkového rámce (viz. cv07):

```
1 push ebp          ;vytvoreni zasobnikoveho ramce
2 mov  ebp, esp     ;vytvoreni zasobnikoveho ramce
```

Funkci ukončujeme zrušením zásobníkového rámce a návratem do volající funkce instrukcí ret:

- Návratem z funkce main ukončíme celý program.

```
3 pop  ebp          ;zruseni zasobnikoveho ramce
4 ret               ;navrat do volajici funkce
```



Nad kódovým segmentem mohou být umístěny ještě datové segmenty (viz. cv03) a nad nimi direktiva %include pro vložení knihovny:

- Knihovna rw32-2018.inc poskytuje funkce pro vstup a výstup programu.
- Najdete ji ve složce SASM -> Windows -> include -> rw32-2018.inc.
- Návod na její použití najdete v knihovně na řádcích 21 až 179.

```
1 %include "rw32-2018.inc" ;vlozeni knihovny pro vstup a vystup
```

Pro volání funkcí používáme instrukci call:

- Názvy funkcí jsou case-sensitive, na psaní malých a velkých písmen záleží.
- Knihovní funkce vždy pracují s registrem EAX, pokud vstup chceme načíst jinam, musíme ho nejprve načíst do EAX a pak přiřadit jinam instrukcí mov.

```
2 call ReadInt32            ;ze vstupu nacti 32b do EAX
3 call WriteInt32           ;na vystup vypis 32b  z EAX
4 call WriteNewLine         ;na vystup vypis prazdny radek
```

Vyzkoušejte si:

- Program který ze vstupu načte dvě čísla (X a Y) a vypíše jejich součet (X+Y).

```
1  %include "rw32-2018.inc" ;vlozeni knihovny pro vstup a vystup
2
3  section .data            ;inicializovany datovy segment
4                           ;zde budou vase data
5
6  section .text            ;zacatek kodoveho segmentu
7  _main:                   ;navesti funkce main, zacatek programu
8      push ebp             ;vytvoreni zasobnikoveho ramce
9      mov  ebp, esp        ;vytvoreni zasobnikoveho ramce
10                          ;                     (EAX = ???, EBX = ???)
11     call ReadInt32       ;nacti hodnotu EAX    (EAX =  X , EBX = ???)
12     mov  ebx, eax        ;EBX = EAX            (EAX =  X , EBX =  X )
13     call ReadInt32       ;nacti hodnotu EAX    (EAX =  Y , EBX =  X )
14     add  eax, ebx        ;EAX = EAX + EBX      (EAX = Y+X, EBX =  X )
15     call WriteInt32      ;vypis hodnotu EAX
16
17     pop  ebp             ;zruseni zasobnikoveho ramce
18     ret                  ;navrat do volajici funkce, konec programu
```

Vyzkoušejte si:

- Napište program který ze vstupu načtěte dvě čísla (X a Y), a pak vypíše hodnoty X-Y, -X+Y a -X-Y, každou na vlastní řádek.

Ladění programu

Vývojové prostředí SASM obsahuje zabudovaný Debugger :

- Podobně jako u vyšších programovacích jazyků, v programu nejprve nastavíme breakpointy a program krokujeme po jednotlivých instrukcích.
- Levé okno ukazuje zdrojový kód a aktuální (příští) instrukci.
- Pravé okno ukazuje aktuální obsah všech registrů.

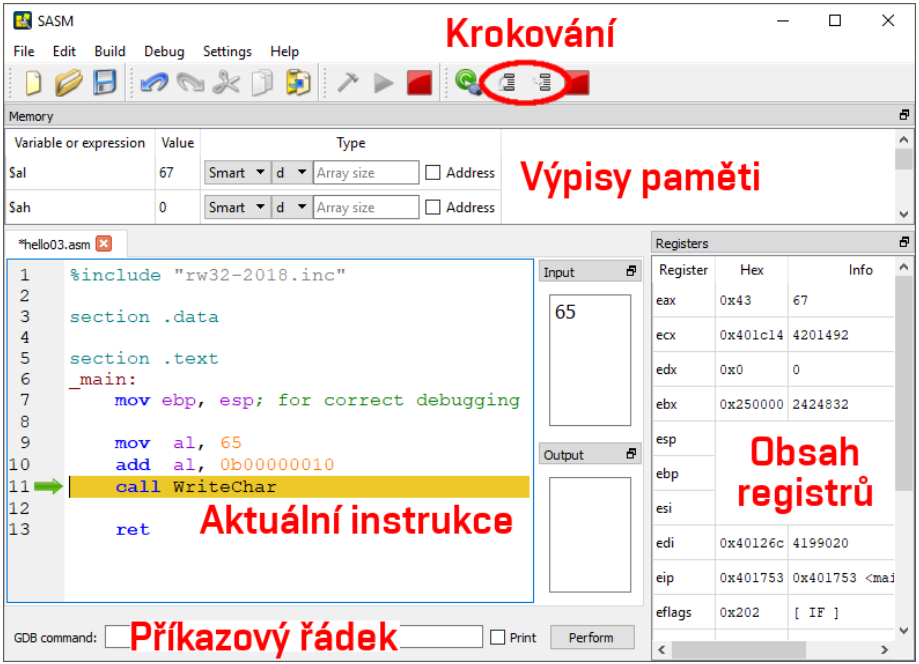
Horní okno slouží k zobrazení obsahu datových segmentů (viz. cv03):

- Umožňuje ale vypsat i obsah jednotlivých registrů a jejich částí.
- Jména musíme psát malými písmeny a s předponou $ ($ah, $al, $ax, $eax).

Výpisům můžeme nastavit různé formátování:

- Smart (automaticky), Hex (šestnáctková), Bin (dvojková), Char (znak ASCII), Int (desítková), Uint (desítková bez znaménka) a Float (desetinné číslo).

V dolní části můžete vidět také příkazovou řádku GDB (nad rámec cvičení ISU).


SASM
File Edit Build Debug Settings Help
Krokování
Memory
Variable or expression
Value
Type
$al
67
Smart
d
Array size
Address
$ah
0
Výpisy paměti
*hello03.asm
1 %include "rw32-2018.inc"
2
3 section .data
4
5 section .text
6 _main:
7 mov ebp, esp; for correct debugging
8
9 mov al, 65
10 add al, 0b00000010
11 call WriteChar
12
13 ret
Aktuální instrukce
Input
65
Output
Registers
Register
Hex
Info
eax 0x43 67
ecx 0x401c14 4201492
edx 0x0 0
ebx 0x250000 2424832
esp
ebp
esi
Obsah registrů
edi 0x40126c 4199020
eip 0x401753 0x401753 <mai
eflags 0x202 [IF]
GDB command:
Příkazový řádek
Print
Perform

Na procvičení

Vyzkoušejte si:

- Napište program který ze vstupu načte tři čísla (X, Y, Z), a pak vypíše hodnoty Y+Z, -X a X-Y-Z, každou na vlastní řádek.

Na procvičení

Vyzkoušejte si:

- Napište program který ze vstupu načte čtyři čísla (X, Y, Z, W), a pak vypíše hodnoty X+W, X-Y a X-Y+Z-W, každou na vlastní řádek.