# Operační systémy 1

## 5. Cvičení

Radek Janoštík

KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

# Úkoly z minula – jak to šlo?

- Kolik se vám toho podařilo udělat?
  1. `int sgn(int i)`
  2. `char max2c(char a, char b)`
  3. `unsigned short min3us(unsigned short a, unsigned short b, unsigned short c)`
  4. `int kladne(int a, int b, int c)`
  5. `int mocnina(int n, unsigned int m)`
  6. Ověření příznaků

- Dalo něco zabrat?

- Nějaké dotazy?

# Úkoly z minula – jak to šlo?

- Kolik se vám toho podařilo udělat?
  1. `int sgn(int i)`
  2. `char max2c(char a, char b)`
  3. `unsigned short min3us(unsigned short a, unsigned short b, unsigned short c)`
  4. `int kladne(int a, int b, int c)`
  5. `int mocnina(int n, unsigned int m)`
  6. Ověření příznaků
- Dalo něco zabrat?
- Nějaké dotazy?
- Prezenčka (až přijdou všichni :) )

# Přístup do paměti

- Dosud jsme se tomu vyhýbali
- Zásadní operace, paměť je ale „dražší“ než registry
- Spojitý, jednotný prostor s $2^{64}$ jednobajtových buněk
  - Adresa od $0$ do $2^{64} - 1$
  - Sousední buňka je „o jedna dál“
  - Větší hodnoty zaberou více buněk za sebou
- Musíme striktně rozlišovat „adresu buňky“ a hodnotu na adrese
- V jazyce C „ukazatel“ a „dereference“

# Adresace paměti

- Mnoho instrukcí umožňuje mít jeden operand místo v paměti
- Zápis `velikost [adresa]`
  - Velikosti: `byte, word, dword, qword`
  - Pro velikosti 1, 2, 4, 8 B
- Adresou je obyčejné celé číslo, může být uloženo v registrech

```
1 mov eax, dword [0x12345678] ; nacte do registru eax hodnotu z
    adresy 0x12345678
2 mov ax, word [rbx]   ; nacte do registru ax hodnotu z adresy,
    ktera je v rbx
3 add al, byte [rbx]   ; pricte k al hodnotu bytu na adrese rbx
4 mov byte [rbx], 0    ; nastavi byte na adrese dane registrem
    rbx na 0
5 add dword [rbx], 2   ; pricte k hodnote na adrese rbx hodnotu 2
```

- U prvních 3 jasné z kontextu (velikost registru) u zbývajících nutno dodat

# Zadání adresy

- Adresu je možné zadávat ve tvaru:

$$adresa = posunuti + baze + index \times factor$$

- Tedy v hranatých závorkách můžeme provádět vypočtení adresy
- *posunuti* – konstanta, *báze a index* – registry, *faktor* – čísla 1,2,4 nebo 8
- Každou z části lze vynechat
- Umožňuje různé módy přístupu do paměti: ukazatel, prvky pole, strukturované datové typy, ...

# Ukazatele

- Ukazatel = adresa (celé číslo) buňky v paměťovém prostoru

```
1  ;;
2  ;; funkce zvysi hodnotu danou ukazatelem o 1
3  ;;
4  ;; void incref(int *n);
5  ;;
6  incref:
7      mov edx, dword [rdi] ; precteme hodnotu do registru edx
           (1. operand obsahuje ukazatel)
8      add edx, 1           ; zvysime hodnotu o 1
9      mov dword [rdi], edx ; ulozime hodnotu zpet
10     ret
```

- Adresa (64bitů) v registru `rdi`, ukazuje na `int`, proto `dword` – 32bitů

# Pole

- Adresa pole = adresa prvního prvku v poli
- Další prvky následují v rozestupech podle velikosti datového typu

```
1  ;; Funkce secte count prvku v poli array.
2  ;;
3  ;; int sum(int count, int *array);
4  ;;
5  sum:
6      mov eax, 0    ; prubezny soucet
7      mov ecx, 0    ; index aktualniho prvku
8  sum_loop:
9      cmp edi, ecx  ; testujeme, zda jsme na konci pole
10     je sum_done   ; ukonceni cyklu
11     add eax, [rsi + rcx * 4] ; pricteni hodnoty do prubezneho
           souctu
12     add ecx, 1    ; prechod na dalsi prvek
13     jmp sum_loop
14 sum_done:
15     ret           ; vraceni vysledku (v eax)
```

# Řetězce

- Práce stejná jako s poli, protože „řetězec je pole znaků ukončený znakem '\n'“

```
1  ;; Replika funkce strcpy ze standardni knihovny.
2  ;; Funkce prekopiruje retezec src do pameti dane ukazatelem dst.
3  ;;
4  ;; void my_strcpy(char *dst, char *src);
5  ;;
6  my_strcpy:
7      mov al, byte [rsi] ; precteme jeden (prvni znak) ze
           zdrojoveho retezce
8      mov byte [rdi], al ; ulozime tento znak do ciloveho retezce
9      cmp al, 0          ; pokud je to znak \0, koncime
10     je done
11     add rdi, 1         ; posuneme se k dalsimu znaku
12     add rsi, 1
13     jmp my_strcpy      ; skok na zacatek cyklu
14 done:
15     ret                ; konec funkce
```

# Strukturované datové typy

- Hodnoty jednotlivých členů struktury jsou uloženy v paměti „za sebou“

```
1 struct foo {
2   int bar;
3   short baz;
4 };
5
6 struct foo qux;
```

- Uložena jako 6 bajtů – 4 bajty pro `bar` a 2 bajty pro `baz`
- Situaci ale komplikuje zarovnání celých struktur (násobky 4 nebo 8 bajtů)
- Zarovnání jednotlivých členů:
  - Jednobajtové hodnoty zarovnány na 1 B
  - Dvoubajtové hodnoty zarovnány na 2 B
  - Čtyřbajtové hodnoty zarovnány na 4 B, osmibajtové na 8 B
- Záleží tedy na pořadí (efektivita využití paměti

# Strukturované datové typy

- Mějme strukturu:

```
struct foo {
  char a;
  short b;
  int c;
};
```

- V assembleru s ní budeme pracovat takto:
  - Mezi `a` a `b` bude jednobajtová mezera

```
;; void do_foo(struct foo *x)
;;
do_foo:
  mov al, [rdi] ; precte hodnotu clenu a do registru al
  mov cx, [rdi + 2] ; precte hodnotu clenu b do registru cx
  mov [rdi + 4], eax ; zapise hodnotu v registru eax do clenu c
  ret
```

# Úkol (1/2)

1. Napište funkci `void swap(int *a, int *b)`, která prohodí hodnoty, které jsou dány ukazateli a a b.

2. Napište funkci `void division(unsigned int x, unsigned int y, unsigned int *result, unsigned int *remainder)`, která celočíselně vydělí hodnotu x hodnotou y a výsledek uloží na místo v paměti dané ukazatelem result a zbytek po dělení uloží do paměti dané ukazatelem remainder.

3. Napište funkci `void countdown(int *values)`, která do pole values uloží posloupnost 10, 9, 8, . . . , 1 (v tomto pořadí).

# Úkol (2/2)

4. Napište funkci `void nasobky(short *multiples, short n)`, která do pole multiples uloží prvních deset násobků čísla n.

5. Napište funkci `int minimum(int count, int *values)`, která vrací nejmenší prvek pole values obsahující count hodnot. Vyzkoušejte, že funkce funguje správně pro kladná i záporná čísla.

6. Napište funkci `unsigned int my_strlen(char *s)`, která se bude chovat jako funkce strlen ze standardní knihovny jazyka C.

7. Napište funkci void `my_strcat(char *dest, char *src)`, která se bude chovat jako funkce strcat ze standardní knihovny jazyka C.