Základní řídící konstrukce

Ing. Dominika Regéciová
Výzkumná skupina formálních modelů

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

iregeciova@fit.vutbr.cz

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Příklad na začátek

- **Hodnota** $(99)_{10}$ **je uložena v registru AL. Převeďte ji do:**
  - binární číselné soustavy (přímý kód)
  - binární číselné soustavy (doplňkový kód)
  - hexadecimální číselné soustavy

Příklad na začátek

- **Hodnota** $(99)_{10}$ **je uložena v registru AL. Převeďte ji do:**
  - binární číselné soustavy (přímý kód): $(01100011)_2$
  - binární číselné soustavy (doplňkový kód):$(01100011)_2$
  - hexadecimální číselné soustavy: $(63)_{16}$

Konvence volání funkcí: M. Jaroš

- Volaný vs. Volající
- Definuje způsob předávání parametrů (registry, zásobník – zprava doleva, zleva doprava).
- Definuje, kdo ze zásobníku uklízí parametry.
- Definuje dekorování jmen symbolů.

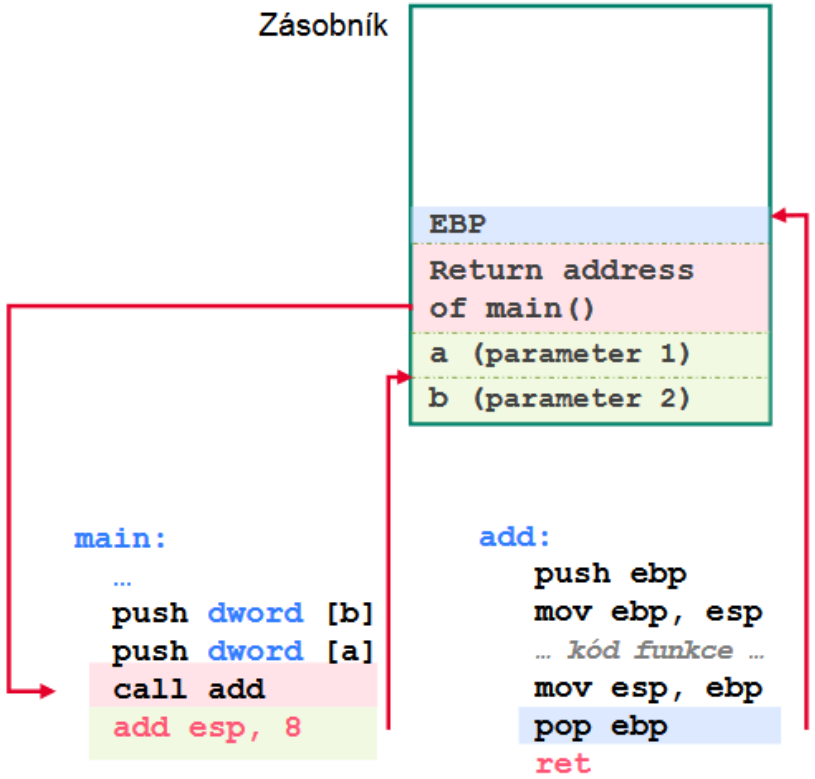
Konvence volání	Parametry funkce	Zásobní k uklízí	Dekorace jmen (pro jazyk C)	Použito v
pascal	zleva doprava	volaný	symbol	Pascal
cdecl	zprava doleva	volající	_symbol	Jazyk C
stdcall	zprava doleva	volaný	_symbol@4	Win42 API
fastcall	první dva parametry v ECX a EDX, zbytek zprava doleva	volaný	@symbol@8	různé

Konvence volání funkcí II: M. Jaroš

```
int add(int x, int y)
{
    return x+y;
}

int main()
{
    int a = 2;
    int b = 5;
    printf('Result: %d\n', add(a, b));
}
```

- cdecl konvence
  - Kdo odstraňuje parametry, které jsme předávali nějaké funkci?
    Volající těsně **za** instrukcí **call**.
  - Jak parametry odstraníme?
    Ručně, posunutím zásobníku na správnou hodnotu.
    Např. `add esp, 8` ; 2 parametry * 4
- stdcall konvence
  - Kdo odstraňuje parametry, které jsme předávali nějaké funkci?
    Volaná funkce.
  - Jak parametry odstraníme?
    Pomocí instrukce RET X ve volané funkci.
    Např. `ret 8` ; 2 parametry * 4




```
main:
…
push dword [b]
push dword [a]
call add
```

```
add:
push ebp
mov ebp, esp
… kód funkce …
mov esp, ebp
pop ebp
ret 8
```

Konstrukce if-else

- `if (cond) statement1; else statement2;`
- Varianta `Jcc`

```
CMP/TEST
Jcc true
(else:)
    statement2
    JMP endif
true:
    statement1
endif:
```

- Varianta `JNcc`

```
CMP/TEST
    JNcc else
(true:)
    statement1
    JMP endif
else:
    statement2
endif:
```

Konstrukce while a do-while

- `while (cond) statement;`

```
while:
    CMP/TEST
    JNcc end
    statement
    JMP while
end:
```

- `do statement while (cond);`

```
dowhile:
    statement
    CMP/TEST
    Jcc dowhile
```

Konstrukce for

- `for (sInit; cond; sUpdate) statement;`
- S využitím `loop`

```
for(ECX=počet; ECX>0; ECX--) statement;

MOV ECX,počet
for:
    statement
LOOP for <- sUpdate je součástí instrukce LOOP (ECX--)
         <- podmínka je součástí instrukce LOOP (ECX>0)
```

- Obecněji


```
for(ECX=počet; ECX>0; ECX--) statement;

MOV ECX,počet
for:
    statement
LOOP for <- sUpdate je součástí instrukce LOOP (ECX--)
         <- podmínka je součástí instrukce LOOP (ECX>0)
```

Konstrukce switch

```
switch (x) {
case val1: statement1; break;
case val2: statement2; break;
. . .
case valN: statementN; break;
default: sDefault;
}
```

```
segment .data
cases DD case_val1
         DD case_val2
         DD case_default
. . .
         DD case_valN
segment .text
    MOV EAX,x
    JMP dword [cases+4*EAX]
case_default:
    sDefault
    JMP break
case_val1:
    statement1
    JMP break
. . .
case_valN:
    statementN
break:
```

Příklady

- Podívejte se na příklady:
  - 8.1.diamant.asm
  - 8.2.switch_example.asm
- Uživatel zadá 2 čísla, vypište to větší (if)
- Uživatel zadá číslo n, zjistěte, zda je dělitelné čísly od 1 do 5 (while)
- Uživatel zadá číslo n, vypište na řádku číslice od 1 do n (for)