



Nízkoúrovňové programování

Služby OS a externí assembler

Petr Krajča



Katedra informatiky
Univerzita Palackého v Olomouci



- registr R7 obsahuje vybrané číslo služby
- seznam `https://chromium.googlesource.com/chromiumos/docs/+master/constants/syscalls.md#arm-32_bit_EABI`
- registry R0 až R5 argumenty, se kterými je služba volána
- služba jádra volána instrukcí SVC #0
- lze potkat i jiné varianty, např. SVC (0x900000 + číslo služby)



```
.globl _start
.text
_start:
    mov r7, #1    @ SYS_EXIT
    mov r0, #42
    svc #0
```

Překlad

```
as -o minp.o minp.asm
ld -o minp minp.o
```



Sekce

- `.text` // obsahuje kód programu
- `.data` // obsahuje měnitelná data
- `.section .rodata` // obsahuje neměnitelná data, konstanty
- `.section .bss` // obsahuje neinicializovaná data

Reprezentace hodnot

- `.byte`, `.short`, `.word` // slouží k přímému vložení hodnot o velikosti (1, 2, 4 B), hodnoty mohou být odděleny čárkou
- `.ascii` // řetězce
- `.space` // vyhradí zadané množství bytů
- `.equ symbol, hodnota` // definuje makro (v kódu nahradí výskyt `symbol` zadanou hodnotou)



```
.globl _start
.text
.equ STDOUT, 1
.equ SYS_EXIT, 1
.equ SYS_WRITE, 4
_start:
    mov r7, #SYS_WRITE
    mov r0, #STDOUT
    adr r1, hello_str
    mov r2, #12
    svc #0
    mov r7, #SYS_EXIT
    mov r0, #42
    svc #0
hello_str:
    .ascii "Hello World\n"
```

- řetězec je uložen v sekci s kódem (.text)
- jeho adresa je určena relativně (instrukce adr)



```
_start:
    mov r7, #SYS_WRITE
    mov r0, #STDOUT
    ldr r1, hello_str_ref
    mov r2, #12
    svc #0
    mov r7, #1 @ EXIT
    mov r0, #42
    svc #0
```

```
hello_str_ref:
    .word hello_str
```

```
.section .rodata
```

```
hello_str:
    .ascii "Hello World\n"
```

- řetezec je uložen v sekci s daty pro čtení (.rodata)
- jeho adresa je získána nepřímým odkazem (instrukce ldr)



- 1 Vytvořte program `rect`, který na terminál vypíše obdélník složený ze znaků '*' o stranách 20×5 .
- 2 Vytvořte program `mypwd`, který se bude chovat podobně jako standardní unixový příkaz `pwd` a vypíše na standardní výstup plnou cestu k aktuálnímu adresáři. Jaký je aktuální adresář zjistíte pomocí systémového volání `getcwd`.
- 3 Vytvořte program `lc`, který vypíše počet řádků přečtených ze standardního vstupu. Náповědy:
 - Program vytvářejte postupně:
 - Spočítejte řádky na vstupu a výsledek vraťte v návratovém kódu.
 - Spočítejte řádky a jejich počet vypíše na standardní výstup.
 - Upravte program, aby pracoval s libovolně velkým vstupem, tj. zpracovával vstup, dokud systémové volání `read` nevrátí 0 nebo zápornou hodnotu.
 - Ještě budete potřebovat: systémové volání pro čtení souborů (`read`), popisovač souboru se standardním vstupem (`stdin`), oblast paměti, kam je možné data načíst (`.bss`)