



Nízkoúrovňové programování

Úvod do instrukční sady ARMv7

Petr Krajča



Katedra informatiky
Univerzita Palackého v Olomouci

- 16 celočíselných (general purpose) registrů (R0–R15) o velikosti 32 bitů
- používá se celý registr
- některé mají vyhrazený význam nebo použití
 - R13 – vrchol zásobníku (SP – stack pointer)
 - R14 – návratová adresa (LR – link register)
 - R15 – adresa v programu (PC – program counter)
- APSR – příznakový registr

Konvence použití registrů

- R0–R3 – předávání argumentů a caller-saved registry
- R4–R8, R10 – callee-saved registry
- R11 – adresa rámce na zásobníku (FP – frame pointer)



`<ins><cond><s> Rd, Rn, imm`

`<ins><cond><s> Rd, Rn, Rm <posun>`

- `<ins>` – název instrukce (např. ADD, MOV, B)
- `<cond>` – podmínka, za které je instrukce vykonána (např. EQ, LT, může být vypuštěno)
- `<s>` – znak S značí, že instrukce má změnit obsah registru APSR podle výsledku operace
- Rd – cílový registr
- Rn, Rm – zdrojový registr
- imm – konstanta (přímá hodnota), konstanty uvozeny # (např. #42)



EQ	Equal
NE	Not equal
CS/HS	Carry Set/Higher or same (unsigned \geq)
CC/LO	Carry Clear/Lower (unsigned $<$)
MI	Negative
PL	Positive or zero
VS	Overflow
VC	No overflow
HI	Higher (unsigned \leq)
LS	Lower or same (unsigned \leq)
GE	Signed \geq
LT	Signed $<$
GT	Signed $>$
LE	Signed \leq
AL	Always (obvykle vypuštěno)



- každá instrukce zakódována 32 bity
- omezený prostor pro uložení konstanty (obvykle vyhrazeno 12 bitů)
- rozsah -2048 do 2047 by nebyl využit efektivně
- konstanty zakodávány jako 8 bitů hodnota (imm) + 4 bity rotace doprava (rr)
- hodnota konstanty je dána jako

$$imm \text{ ROR } (rr \times 2)$$

- např. hodnoty 000000XY nebo 00XY0000

přiřazení 32bitové hodnoty do registru

- instrukce MOVW Rd, imm16 (lze použít jen MOV) nastaví spodních 16 bitů registru Rd a vynuluje horních 16 bitů
- instrukce MOVT Rd, imm16 nastaví horních 16 bitů registru Rd

název	popis	význam
ADC Rd, Rn, Op2	add with carry	$Rd = Rn + Op2 + \text{carry}$
ADD Rd, Rn, Op2	add	$Rd = Rn + Op2$
MOV Rd, Op2	move	$Rd = Op2$
MVN Rd, Op2	move NOT	$Rd = \sim Op2$
RSB Rd, Rn, Op2	reverse subtract	$Rd = Op2 - Rn$
RSC Rd, Rn, Op2	reverse subtract with carry	$Rd = Op2 - Rn - !\text{carry}$
SBC Rd, Rn, Op2	subtract with carry	$Rd = Rn - Op2 - !\text{carry}$
SUB Rd, Rn, Op2	subtract	$Rd = Rn - Op2$
AND Rd, Rn, Op2	AND	$Rd = Rn \& Op2$
BIC Rd, Rn, Op2	bit clear	$Rd = Rn \& \sim Op2$
EOR Rd, Rn, Op2	exclusive OR	$Rd = Rn \wedge Op2$
ORR Rd, Rn, Op2	OR	$Rd = Rn Op2$
MUL Rd, Rn, Op2	multiply	$Rd = Rn * Op2$
SDIV Rd, Rn, Op2	signed division	$Rd = Rn / Op2$
UDIV Rd, Rn, Op2	unsigned division	$Rd = Rn / Op2$



Porovnání

název	popis	význam
CMP R_n, Op_2	compare	$R_n - Op_2$
CMN R_n, Op_2	compare negative	$R_n + Op_2$
TEQ R_n, Op_2	test EQuivalence	$R_n \hat{=} Op_2$
TST R_n, Op_2	test	$R_n \& Op_2$

Skoky

- B – skok na zadanou adresu
- BX – skok s možným přechodem na jiný typ kódování instrukcí (Thumb)
- podmíněné skoky získáváme zdarma díky podmíněnému provádění instrukcí
- BX LR – slouží k návratu z funkce (LR obsahuje návratovou adresu)
- alternativně lze měnit obsah registru PC pomocí aritmetických operací (např. ADD), opatrně, nutné konzultovat se specifikací, v řadě případů je chování nedefinované (unpredictable)



```
.global rect_perimeter

/*
 * obvod ctverce
 * int rect_perimeter(int, int)
 */
rect_perimeter:
    add r0, r0, r1 @ soucet stran
    add r0, r0, r0 @ krat dva
    bx lr
```

■ `as -o foo.o foo.asm`

absolutní hodnota (s využitím skoků)

abs:

```
cmp r0, #0      @ porovnani argumentu s 0
bge abs_done   @ pokud je nezaporna, konec
neg r0, r0     @ otoceni znamenska
```

abs_done:

```
bx lr
```

absolutní hodnota (bez skoků)

abs:

```
cmp r0, #0      @ porovnani argumentu s 0
neglt r0, r0   @ otoceni znamenska, pokud je hodnota mensi
bx lr
```

- instrukce NEG Rd, Rn je alias pro RSB Rd, Rn, 0



minimum dvou hodnot (s využitím skoků)

```
min:
    cmp r0, r1    @ porovnani argumentu
    blt min_done @ r0 jiz obsahuje mensi hodnotu
    mov r0, r1    @ r1 je mensi, proto ji priradime do r0
min_done:
    bx lr
```

minimum dvou hodnot (bez skoků)

```
min:
    cmp r0, r1    @ porovnani argumentu
    movgt r0, r1  @ r0 je vetsi, proto do r0 priradime r1
    bx lr
```



- ARM Cortex-A Series Programmer's Guide for ARMv7-A
<https://developer.arm.com/documentation/den0013/latest/>
- ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition
<https://developer.arm.com/documentation/ddi0406/latest/>
- Procedure Call Standard for the Arm Architecture:
<https://github.com/ARM-software/abi-aa/releases/download/2022Q3/aapcs32.pdf>



- Napište funkci `int` `obvod_trojuhelnika(int a, int b, int c)`, která spočítá obvod trojúhelníka.
- Napište funkci `int` `objem_krychle(int a)`, která spočítá objem krychle.
- Napište funkci `int` `avg(int a, int b, int c)` pro výpočet aritmetického průměru tří čísel typu `int`.
- Napište funkci `int` `sgn(int i)`, která vrací hodnoty -1, 0, 1 v závislosti na tom, zda-li je hodnota `i` záporná, nulová nebo kladná.
- Napište funkci `unsigned short` `min3u(unsigned int a, unsigned int b, unsigned int c)`, která vrátí minimum ze tří zadaných hodnot.
- Napište funkci `int` `fact(int n)`, která vrátí faktoriál čísla `n`.
- Napište funkci `int` `mocnina(int n, unsigned int m)`, která vrátí hodnotu čísla `n` umocněnou číslem `m`.

Poznámka: Vhodně použijte podmíněné skoky a podmíněné vykonávání instrukcí.