

**Západočeská univerzita v Plzni**  
**Fakulta aplikovaných věd**

**Semestrální práce z předmětu POT**  
**Jednoduchá kalkulačka**

Dominik Zappe (A20B0279P)

13. 4. 2021

## Zadání úlohy

Provádí operace +, −, ×, / s čísly int16.

Zadá se např. 25+66=, 33−12=, 18×6=, 77/7=.

Program zobrazí výsledek a čeká na další zadání.

## Popis algoritmu

V paměti na adresách *cis1*, *cis2*, *znamenko* a *vysledek* jsou vymezeny příslušně velké mezery. (*cis1* a *cis2* mají 2 byty, protože se očekává maximálně 16 bitů, ze zadání int16; na *znamenko* stačí 1 byte; *vysledek* má vymezených 5 bytů, 4 na výsledek a pátý pro odřádkování). Dále je také v paměti vymezeno 20 bytů pro *vstup* (reálně by stačilo 10, ale rezervoval jsem více).

Očekáván je vstup, jak bylo uvedeno v zadání, tj. „*cis1znamenkocis2*=“. Tento vstup se pak rozkouskuje podle znaménka na dvě čísla – *cis1* a *cis2* a také se uloží „hodnota“ operace mezi nimi na adresu *znamenko*. Symbolicky jsem si řekl, že sčítání bude 1, odčítání 2, násobení 3 a dělení 4. Čísla je také zapotřebí převést z ascii na číselné hodnoty (1-9 a A-F).

Když už jsou čísla načtená a je známá i požadovaná operace, vykoná se daná operace a výsledek se musí zpětně převést z čísla na ascii. Postupně se tyto ascii znaky ukládají do paměti na adresu *vysledek* a nakonec se jako 5. byte vloží 0x0A (odřádkování).

Všechny inputy a outputy se dějí pomocí GETS a PUTS a syscall (0x1FF00).

Program vypíše tento výsledek a podmíněným skokem se vrací zpět na začátek a vyžádá si další zadání příkladu.

## Podprogramy (a jejich detailnější popis algoritmu)

### vynulujReg

Podprogram vynuluje všechny (až na ER7) registry za pomoci xor.l (např. xor.l ER0,ER0).

### nacti

Podprogram je určený pro správné načtení vstupních dat. Nejprve se zkopíruje adresa *vstupu* do registru ER6. Pak se vezme jeden byte z ER6 a vloží se do R0L. Následně se testuje, jestli tento byte neodpovídá znakům + − × / nebo =, tím by se podprogram přerušil. První číslo se tedy čte, dokud se nenarazí na znaménko operace, jeho symbolická „hodnota“ se ukládá průběžně do registru R5L (1 symbolizuje sčítání, 2 odčítání, 3 násobení a 4 dělení). Od načteného bytu se odečte hodnota ascii 0 (0x30) a porovná se s 9. Je-li menší nebo roven, jedná se o číslo 0-9, není-li tomu tak, musí se ještě přičíst konstanta ascii 0 mínus ascii A plus 0x0A. Tím se převede znak na A-F. Bitově se tento znak posune a jde se načíst další byte (posun pointeru v ER6). V případě, že se našlo znaménko, je jeho symbolická hodnota uložená v R5L a před ukončením programu se číslo bitově

posune doprava, aby bylo správně zarovnané, a posune se pointer v ER6 o jedna, aby se mohlo jít načítat další číslo.

Při načítání druhého čísla se děje to samé, jen na konci podprogramu není relevantní obsah R5L, protože znaménko už načtené je.

## vypis

Podprogram je určený pro výpis výsledku do simulovaného outputu. Výsledek výpočtu je v registru R0. Tam se po celou dobu drží jeho originální hodnota. Tato hodnota se zkopíruje do R1 a provede se bitový posun doprava tak, aby nejvyšší (4.) řád byl na posledním bytu registru. Přičte se ascii 0 (0x30) a opět se ověřuje, jestli se jedná o číslo 1–9 (0x30 – 0x39) nebo o A–F. Když se nejedná o číslo 1–9 musí se ještě přičíst konstanta ascii A mínus ascii 0 mínus 0x0A. Obsah tohoto registru se nakopíruje na adresu *vysledek*.

Znovu se zkopíruje registr R0 do registru R1 a tentokrát se provede bitový posun tak, aby v registru zbyly 2 nejvyšší řády. Je zapotřebí se zbavit řádu vyššího, aby v registru zbylo jen 0x (kde x značí číslo od 0 do F). To se provede tak, že se R1L zkopíruje do R2L a R2 se posune doprava a doleva, zbaví se tak nižšího řádu. Toto číslo se pak odečte od R1L a v R1L už zbude požadované číslo. To se opět převede na ascii a zapíše se na adresu *vysledek+1*.

Obdobným způsobem se řeší ještě další 2 cifry a jako poslední pátý znak se vloží na adresu *vysledek+4* znak odřádkování – 0x0A.

## Použité proměnné

<b>vstup</b>	- vstup od uživatele
<b>vyzva</b>	- výzva "Zadej příklad: "
<b>odpoved</b>	- odpověď "Vysledek je: ", za tento řetězec se vkládá reálný výsledek
<b>cis1</b>	- první číslo v příkladu
<b>cis2</b>	- druhé číslo v příkladu
<b>vysledek</b>	- výsledek příkladu
<b>znamenko</b>	- požadovaná operace v příkladu

<b>par_vyzva</b>	- parametrický blok, ukazatel na <b>vyzva</b>
<b>par_vstup</b>	- parametrický blok, ukazatel na <b>vstup</b>
<b>par_vystup</b>	- parametrický blok, ukazatel na <b>vystup</b>
<b>par_odpo</b>	- parametrický blok, ukazatel na <b>odpoved</b>
<b>stck</b>	- zásobník, nastaven na velikost 100B

## Proměnné v paměti

Proměnná	Adresa v paměti
vstup	FF4000
vyzva	FF4014
odpoved	FF4024
cis1	FF4032
cis2	FF4034
vysledek	FF4036
znamenko	FF403B
par_vyzva	FF403C
par_vstup	FF4040
par_vystup	FF4044
par_odpo	FF4048
stck	FF40B0

## Konkrétní obsah registru SP v podprogramu

Konkrétní příklad bude uveden na prvním volání podprogramu

41				
42	000422		jsr	vynulujReg
43	000426		mov.l	#vstup,ER6
44	00042C		lrr	nacni

Na obrázku je vidět, že vstup do podprogramu je na symbolické adrese 000422, po výstupu z podprogramu tedy musí program pokračovat adresou 000426.

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	ASCII
FF4000	31	2B	31	3D	0A	00	00	00	00	00	00	00	00	00	00	00	l+l=.....
FF4010	00	00	00	00	5A	61	64	65	6A	20	70	72	69	6B	6C	61	....Zadej prikla
FF4020	64	3A	20	00	56	79	73	6C	65	64	65	6B	20	6A	65	3A	d: .Vysledek je:
FF4030	20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	FF 40 14	.....@.
FF4040	00	FF	40	00	00	FF	40	36	00	FF	40	24	00	00	00	00	..@...@6..@\$....
FF4050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
FF4060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
FF4070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
FF4080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
FF4090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
FF40A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	04 26	.....s
FF40B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

  

ER6	00000000
ER7	00FF40AC
PC	000484

ER7 (SP) ukazuje na adresu v paměti 00FF40AC, za kterou je červeně vidět, že je tam uložena adresa 000426. Voláním *rts* se přesuneme zpět do hlavního programu na tuto adresu.

Obdobně (dle očekávání) to funguje pro všechny ostatní volání podprogramů a zásobník funguje správně.