Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Semestrální práce z předmětu POT

Vypracovala: Dudchuk Olesya A17B0638P

Datum: 17.05.2018

1. Zadání úlohy

Program převede číslo int32 na float.

2. Popis algoritmu řešení

Program je rozdělen na tři části :
1) inicializace proměnných, jejich naplnění, příprava běhu programu
2) načitaní čísla z klávesnice
3) převod čísla int32 na float a výpis

3. Popis programu

3.1 Použité proměnné

Program při svém běhu využívá čtří proměnné uložené v datovém segmentu DATA a sadu registrů.

datový segment DATA:

stck – zasobnik
buffer – vstupní řetezec
text – text "Zadejte cislo:\n"
out – vystupní řetezec

3.2 Hlavní program

Hlavní část programu začíná návěštím START. Inicializuje se zasobnik. Do ER1 se ulozi adresa zacatku vety *"Zadejte cislo:\n"*. Dal jde načitaní čisla s klávesnicy při pomoci procedury *loop_asci2decimal* do registru ER4. Do ER1 se načitá adresa vystupního bufferu. Do R2H umisten počet čislic. Posuneme nejvyšší hexidecemalné čislo do prava. Do R2L zadáme masku na hexidecemalné čislo. Převedeme čislice 0 až 9 do hex. soustavy. Když bulo čislo větší než 9, uděláme opravu na A až F. Číslo uložím do bufferu. Posuneme pointer a odečteme od počtu čislic edničku. Opakujeme ještě osmkrát (pokud R1L a R1H nejsou stejné). Ukončíme a vypišeme text a decimalní číslo . Na konci programu se spusti nekonecna smycka.

3.3 Procedura

procedura loop_asci2decimal:

Funkce prevede ASCII řetezec na čislo
vstup: ER0 = adresa řetězce; vystup: R4 = čislo

Uložíme registr. Vynulujeme registry ER4 a ER5. Přečteme znak z paměti. Uděláme test na znak ,-, , pokud je záporné, do ER5 se vloží hodnota 0xFFFFFFFF, kterou používáme při převodu. Odečteme 30, aby převest do čisla. Zvětšíme čislo v ER4 desetkrat a přidáme načtene čislo. Pak smyčka na další čislo. Obnovíme registr na puvodni hodnotu a udeláme skok na proceduru *convert*, kdy R1L a R2L jsou stejné.

3.4 Popis algoritmu

Převod se uděla bitovym posunem vstupu, který je uložen jako výsledek operaci *and.l* ER4(vstup zadaný uživatelem) a ER0(hodnota 0x7FFF), a je uložen do ER0, a snižením hodnoty exponentu (ER1 = 141), dokud první 1 dosahne bitovou pozici 23. Výsledek převodu je uložen do ER6.

4. Obsluha Programu

Program je napsaný ve zdrojovém textu pro procesor H8S, přeložitelný a spustitelný na emulátoru procesoru HEW pro Windows.

Přiklad vstupu a výstupu:

Zadejte vstup:
45
Vystup: 42340000

5. Závěr

Program splňuje zadání a nebyli zjištěny žádné chyby.

6. Kod programu

```
; Program prevede int32 na float
;
        .h8300s
      .equ    syscall,0x1FF00    ; simulovany vstup/vystup
      .equ    PUTS,0x0114              ; kod PUTS
                .equ    GETS,0x0113       ; kod GETS
                .equ    PUTC,0x0112       ; kod PUTC
;-------datovy segment----------------------------------------------------------
      .data

buffer: .space 100              ;vstupni buffer
text:    .asciz  "Zadejte cislo:\n"
text2:  .asciz  "Vystup: "
out:    .space 20               ;vystupni buffer
                .align  2               ;zarovnani parametrickeho bloku
p_buffer: .long buffer          ;parametricky blok pro vstup dekadickeho cisla
p_vt:     .long text            ;parametricky blok pro vystup textu
p_tx:     .long text2           ;parametricky blok pro vystup textu
p_out:   .long out              ;parametricky blok pro vystup cisla float

        .align 1                ;zarovnani adresy
        .space 100              ;stack
```

```
stck:
;---------kodovy segment-------------------------------------------------------
        .text
        .global _start

;---------hlavni algoritm programu, prevede int32 na float-------------------------

result_0:
        mov.l #0x0, ER6          ;vysledek = 0

count_result:
        shal.l #2, ER1           ;bitovy posun doleva(23 bity)
        shal.l #2, ER1
        shal.l #2, ER1
        shal.l #2, ER1
        shal.l #2, ER1
        shal.l #2, ER1
        shal.l #2, ER1
        shal.l #2, ER1
        shal.l #2, ER1
        shal.l #2, ER1
        shal.l #2, ER1
        shal.l #2, ER1
        shal.l  ER1

        mov.l #0x7FFFFF, ER2     ;
        and.l ER0, ER2
        or.l ER1, ER2
        mov.l ER2, ER6

exit:                        ; navrat z podprogramu
        rts
declare_var:
        shal.l   ER0
        dec.l #1, ER1
```

```
        jmp @continue_for_cyklus

convert_positive:
        mov.l #0x7FFF, ER0
        and.l ER4, ER0                  ; fraction = input &I2F_MAX_INPUT
        mov.l #0x0, ER1
        cmp.l ER0, ER1
        beq result_0

        mov.l #141, ER1             ;exponent

        shal.l #2, ER0
        shal.l #2, ER0
        shal.l #2, ER0
        shal.l #2, ER0
        shal.l ER0

        mov.l #15, ER2
        mov.l #0, ER3

for_cyklus:
        mov.l #0x800000, ER5
        and.l ER0, ER5
        cmp.l ER5, ER3 ;if ER5 is 0
        beq declare_var
        jmp @count_result

continue_for_cyklus:
        cmp.l ER2, ER3
        beq exit
        dec.l #1, ER2
        jmp @for_cyklus

convert_negative:
        xor.l ER5, ER4
```

```
        jsr @convert_positive
        mov.l #0x80000000, ER1
        or.l ER1, ER6
        jmp @print_result

convert:
        xor.l ER5, ER4
        mov.l #0, ER6           ;kontroluje jestli je vstup < 0
        cmp.l ER6, ER4
        bmi convert_negative    ;pokud vstup je < 0, skok na "convert_negative"
        jsr @convert_positive   ;jinak provede funkci "convert_positive"
        jmp @print_result

negative_num:
        mov.l #0xFFFFFFFF, ER5
        inc.l #1, ER0
        jmp @loop_asci2decimal
;----------zacatek programu-----------------------------------------------------

_start:
        mov.l #stck, ER7
        mov.w #PUTS,R0              ; 24bitovy PUTS
        mov.l #p_vt,ER1             ; adr. param. bloku do ER1
        jsr @syscall
;———cteni cisla
————————————————————————————————————————————————————————————————————
        mov.w #GETS,R0              ;24bitovy GETS
        mov.l #p_buffer,ER1  ;adr. param. bloku do ER1
        jsr @syscall

        mov.l @p_buffer, ER0
        mov.l #0, ER4               ;vynulovani
        mov.l #0, ER5           ;vynulovani
;----------prevede ascii na decimalni cislo-----------------------------------
```

```
loop_asci2decimal:
        mov.b @ER0, R1L
        mov.b #0x2D, R2L
        cmp.b R1L, R2L
        beq negative_num
        mov.b #0x0A, R2L
        cmp.b R1L, R2L
        beq convert             ;skok na prevod cisla
        mov.b #0x30, R1H
        sub.b R1H, R1L
        mov.l #0x000000FF, ER3
        and.l ER1, ER3
        mov.w #0xA, E2
        mulxs.w E2, ER4
        add.l ER3, ER4
        inc.l #1, ER0
        jmp @loop_asci2decimal
;--------funkce prevodu do ascii kodu---------------------------------------
print_char:
        add.b #0x37, R0H
        mov.b #0, R0L
        push.w R0
        jmp @print_end
print_number:
        add.b #0x30, R0H
        mov.b #0, R0L
        push.w R0
        jmp @print_end
print_result:
        mov.b #8, R1L
        mov.b #0, R1H
print_loop:
        cmp.b R1L, R1H
        beq outprint_preloop
        mov.l #0x0000000F, ER0
```

```
        and.l ER6, ER0                  ;maska na posledni 4 bity
        mov.b R0L, R0H
        mov.b #9, R0L
        cmp.b R0L, R0H
        ble print_number
        jmp @print_char
print_end:
        shlr.l #2, ER6
        shlr.l #2, ER6
        inc.b R1H
        jmp @print_loop
---------vypis do konzole-----------------------------------------------------------
outprint_preloop:
        mov.b #8, R1L
        mov.b #0, R1H
        mov.l #out, ER3

outprint_loop:
        cmp.b R1L, R1H
        beq outprint
        pop.w R2
        mov.b R2H, @ER3
        inc.l #1, ER3
        inc.b R1H
        jmp @outprint_loop
outprint:
        mov.l #p_out, ER1
        mov.w #PUTS, R0
        jsr @syscall
;------konak programu-------------------------------------------------------
 konec: jmp @konec
      .end
```