Bajczi Levente
XAO5ER
Előadó/gyakorlatvezető/laborvezető: Renczes Balász

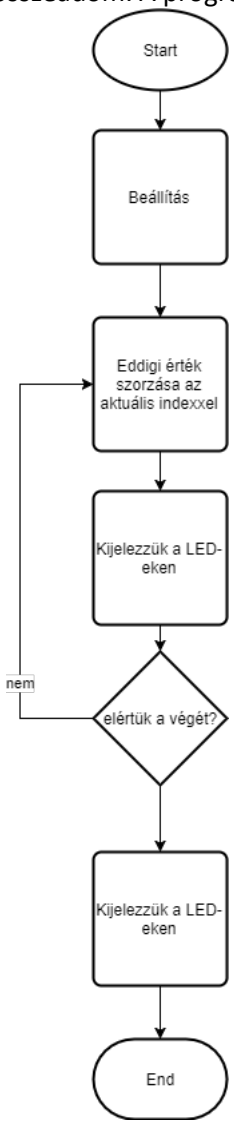
Faktoriális generátor program MiniRISC-re, érték a memóriában BCD-ben, amíg > GOOGOL! (70!)

A feladat megvalósításában a lehető legegyszerűbb módszert választottam: f(0) = 1, és minden f(n) = f(n-1) * n.

Mivel az utasításkészlet nem tartalmaz szorzást, főleg nem ekkorát, más módszert kellett kitalálnom. A megoldás, amit kitaláltam, abból áll, hogy egy háromszorosan egymásba ágyazott ciklusban összeadok, aminek a vége egy nagyszámokkal történő szorzás lesz. Ehhez először lemásolom az aktuális értéket a memória második feléből az első felébe, majd ezt az értéket adom hozzá az eredeti helyen lévő adatokhoz n-szer minden ciklusban. Ez egy nagyon nem optimális megoldás, sokféleképpen lehetne optimalizálni – ebből az én megoldásom csak azt tartalmazza, hogy minden n>10 számra először 4 bittel balra shifteli az egész értéket, amivel egy helyiértékkel nőtt minden BCD számjegy, tehát egy 10-zel való szorzást valósítottam meg.

Ez a megoldás maximum 84! < 10^(0x40*2) < 85! számokat képes megjeleníteni, tehát max 84!-ig mehetünk.

A programom moduláris, azaz a moduljait egyesével is lehet használni – ilyen pl. az a függvény, ami lenullázza a legelején a teljes memóriát (ezt nem tehetjük alapból fel, hogy 0), vagy ilyen a BCD összeadóm. A program fő váza a következő modell szerint működik:



A teljes program vázának részletezése ASM modellel túl hosszú lenne, ezért csak szóban fejtem ki (illetve a programban szinte minden sor végén van comment, hogy épp mint tettem az adott sorban):

Beállítás

- Felhasznált regiszterek lenullázása (XOR önmagával)
- Memóriaterület lenullázása
- Kezdőérték (0! = 1) memóriába írása

Szorzás

- Adott értéket a memória felső részébe átmásoljuk
- Eliminájuk az első tíz összeadást (ha tudjuk)
- Hozzáadjuk a lemásolt értéket az eredetihez n-szer
  - *addbcd* segítségével az egész alsó 0x40 byte-ot adjuk a teljes felső 0x40 byte-hoz

LED

- Megszámoljuk a tízeseket
- Levonjuk a tízesek számát (*10) az egyesekből
- Össze VAGY-oljuk a két értéket
- Kiírjuk ezt a LED-ekre

Végérték

- Definiálva van (alapból 84, ez a legnagyobb érték)

LED

- Megszámoljuk a tízeseket
- Levonjuk a tízesek számát (*10) az egyesekből
- Össze VAGY-oljuk a két értéket
- Kiírjuk ezt a LED-ekre, ekkor olvassuk ki a memória felső 40 byte-ját

End

- Tartjuk az értéket

A kód maga (.lst file): GitHub Link

Bajczi Levente
XAO5ER
Előadó/gyakorlatvezető/laborvezető: Renczes Balász

```
1   LOGSYS MiniRISC v2.0 assembler v1.0 list file
2   Copyright (C) 2013 LOGSYS, Tamas Raikovich
3   Source file: fakultaet-1.s
4   Created on : 08/12/2017 21:05:26
5
6   S Addr  Instr   Source code
7   ------------------------------------------------------------------------------------------
8                   DEF SIZE 0x80
9                   DEF HALF 0x40
10                  DEF LD   0x80
11                  DEF END  0x54
12                  ;A feladatunk tulajdonkeppen egy 100-as szamrendszerben levo kaszkadositott osszeado realizalasa
13
14                      CODE
15  C 00    F060        xor     r0, r0                  ;lenullazzuk a szamlalot
16  C 01    F161        xor     r1, r1                  ;lenullazzuk a segedregisztert
17  C 02    C280        mov     r2, #SIZE[80]           ;betoltjuk r2-be a kinullazando terulet meretet
18  C 03                null:                           ;Memoriaterulet nullazasa, hogy ne legyen benne szemet
19  C 03    F190        mov     (r0), r1                ;az r0 altal jelzett helyre r1 erteket (0) kiirni
20  C 04    0001        add     r0, #0x01               ;a mutatot noveljuk
21  C 05    F0A2        cmp     r0, r2                  ;teszteljuk, hogy elertuk-e a veget
22  C 06    B203        jnz     null[03]                ;ha nem, akkor visszamegyunk a loop elejere
23  C 07                _start:                         ;itt kezdodik a valos program
24  C 07    C101        mov     r1, #0x01               ;a segedregiszterbe 1-et toltunk, ami a kezdoertek
25  C 08    C080        mov     r0, #SIZE[80]           ;
26  C 09    2001        sub     r0, #0x01               ;az utols byte-ra allitjuk a mutatot
27  C 0A    F190        mov     (r0), r1                ;Kezdoertek, 0! = 1 (definicio alapjan)
28
29  C 0B    FD6D        xor     r13, r13                ;lenullazzuk a carryt
30
31  C 0C    F060        xor     r0, r0                  ;lenullazzuk a mutatot
32  C 0D                loop:
```

```
33  C 0D    B939        jsr     copy_second_half[39]    ;a memoria masodik felet lemasoljuk az elso felebe
34  C 0E    F161        xor     r1, r1                  ;lenullazzuk a segedregisztert, ebben fogunk a szubrutinban szamolni felfele
35  C 0F    F8C0        mov     r8, r0                  ;elmentjuk r0 ertelet
36  C 10    0801        add     r8, #0x01               ;noveljuk ezt az erteket (igy tudjuk hasznalni, ld. tizesek)
37  C 11    B923        jsr     tizesek[23]             ;a tizeseket eliminaljuk, hogy joval kevesebb szamitast kelljen veghezvinni
38  C 12    B942        jsr     inner_loop[42]          ;a belso loop, ami arra van, hogy az osszeadasokat elvegezze
39  C 13    0001        add     r0, #0x01               ;noveljuk a szamlalo erteket
40  C 14    B918        jsr     display_led[18]         ;kijelezzuk a LED-eken (BCD)
41  C 15    A054        cmp     r0, #END[54]            ;ha elertuk a definialt vegerteket (END<=83)
42  C 16    B20D        jnz     loop[0D]                ;akkor mar nem terunk vissza a loop elejere, amugy igen
43  C 17            end:
44  C 17    B017        jmp     end[17]                 ;megtartjuk ezt az allapotot, elertuk a vegerteket
45
46
47  C 18            display_led:
48  C 18    F8C0        mov     r8, r0                  ;elmentjuk r0 erteket, hogy azt ne bantsuk
49  C 19    F969        xor     r9, r9                  ;lenullazzuk a segedregisztert, amiben a BCD kod fog eloallni
50  C 1A            display_loop:
51  C 1A    A80A        cmp     r8, #0x0A               ;ha mar az ertek kisebb, mint 10, akkor vegeztunk,
52  C 1B    B51F        jn      display[1F]             ;akkor mehetunk a display label-re
53  C 1C    0901        add     r9, #0x01               ;noveljuk a tizesek szamat (majd SWP-zzük)
54  C 1D    280A        sub     r8, #0x0A               ;levonunk tizet a szambol
55  C 1E    B01A        jmp     display_loop[1A]        ;vissza a loop elejere
56  C 1F            display:
57  C 1F    7900        swp     r9                      ;felcsereljuk az also es a felso 4 bitet, ezzel egy helyiertekkel feljebb kerul az ertekes resz
58  C 20    F859        or      r8, r9                  ;ossze VAGY-oljuk az also es a felso digitet
59  C 21    9080        mov     LD[80], r8              ;kiirjuk ezt a LED sorra
60  C 22    BA00        rts
61
62  C 23            tizesek:
63  C 23    A80A        cmp     r8, #0x0A               ;ha hozzaadunk tizet, es ez az ertek nagyobb, mint r0, akkor visszaterunk, nem csinalunk mar semmit
64  C 24    B529        jn      ret_tizesek[29]         ;
```

```
65  C 25    B92A        jsr     shift_left[2A]          ;ha van benne egesz tizes, balra shifteljuk fel byte-tal
66  C 26    280A        sub     r8, #0x0A               ;levonunk tizet az ertekebol
67  C 27    F1C0        mov     r1, r0                  ;betoltjuk r1-be r0 erteket,
68  C 28    F128        sub     r1, r8                  ;makd kivonjuk belole a maradekot - ennyi osszeadast kell elvegezni a tizes szorzas utan
69  C 29            ret_tizesek:
70  C 29    BA00        rts                             ;
71
72  C 2A            shift_left:
73  C 2A    FD6D        xor     r13, r13                ;lenullazzuk a carry regisztert
74  C 2B    C380        mov     r3, #SIZE[80]           ;betoltjuk az utolso byte cimet (+1)
75  C 2C            shift_left_loop:
76  C 2C    2301        sub     r3, #0x01               ;a mutatot egyel visszaallitjuk
77  C 2D    A340        cmp     r3, #HALF[40]           ;ha mar a felenel lejjebb tartunk, kilepunk a loopbol, vegeztunk
78  C 2E    B538        jn      ret_shift_left[38]      ;ha elerjuk az also felet a memorianak, akkor abbahagyjuk
79  C 2F    FCD3        mov     r12, (r3)               ;betoltjuk az aktualis byte-ot
80  C 30    7C00        swp     r12                     ;megcsereljuk az also es felso 4 bitet
81  C 31    FECC        mov     r14, r12                ;elmentjuk r14-ben ezt az erteket
82  C 32    4CF0        and     r12, #0xF0              ;also negy byte minket nem erdekel
83  C 33    FC5D        or      r12, r13                ;osszevagyoljuk a carryvel, amirol tudjuk, hogy negy bit van benne, raadasul alul
84  C 34    FC93        mov     (r3), r12               ;visszairjuk a memoriaba
85  C 35    4E0F        and     r14, #0x0F              ;az elmentett beolvasott felso 4 bitjet tartjuk meg (swp utan vagyunk)
86  C 36    FDCE        mov     r13, r14                ;es ezt beleirjuk a carrybe
87  C 37    B02C        jmp     shift_left_loop[2C]
88
89  C 38            ret_shift_left:
90  C 38    BA00        rts
91
92  C 39            copy_second_half:
93  C 39    F363        xor     r3, r3                  ;lenullazzuk a mutatot
94  C 3A            copy:                               ;Memoriaterulet lemasolasa
95  C 3A    F4C3        mov     r4, r3                  ;lemasoljuk az eppen aktualis cimet
96  C 3B    0440        add     r4, #HALF[40]           ;hozzaadjuk a forras cimehez a memoria felenek nagysagat
```

Bajczi Levente
XAO5ER
Előadó/gyakorlatvezető/laborvezető: Renczes Balász

```
 97  C 3C    F1D4        mov     r1, (r4)                ;betoltjuk a forras erteket
 98  C 3D    F193        mov     (r3), r1                ;kiirjuk a celba
 99  C 3E    0301        add     r3, #0x01               ;noveljuk a mutatot
100  C 3F    A340        cmp     r3, #HALF[40]           ;ha a felenel tartanank, vegeztunk
101  C 40    B23A        jnz     copy[3A]
102  C 41    BA00        rts
103
104  C 42            inner_loop:                         ;tudjuk, hogy r0: vegertek, r1: szamoloregiszter
105  C 42    F0A1        cmp     r0, r1                  ;amig nem lesz r0=r1, addig osszeadjuk az eredeti szamot (amit lemasoltunk) onmagahoz
106  C 43    B148        jz      return[48]
107  C 44    0101        add     r1, #0x01               ;noveljuk a szamoloregisztert
108  C 45    C701        mov     r7, #0x01               ;ezt elmentjuk az r7-ben
109  C 46    B949        jsr     inner_inner_loop[49]    ;a belso-belso fgv meghivasa
110  C 47    B042        jmp     inner_loop[42]          ;elejerol ismet
111
112  C 48            return:
113  C 48    BA00        rts                             ;visszaterunk a szubrutinbol
114
115  C 49            inner_inner_loop:                   ;az osszeadas megvalositasa-a teljes felso felehez hozzaadjuk a teljes also felet-hasonlit a lemasolashoz
116  C 49    C280        mov     r2, #SIZE[80]
117  C 4A    F227        sub     r2, r7
118  C 4B    FED2        mov     r14, (r2)
119  C 4C    F5C2        mov     r5, r2
120  C 4D    2540        sub     r5, #HALF[40]
121  C 4E    FFD5        mov     r15, (r5)
122  C 4F    B956        jsr     addbcd[56]              ;segedfgv, ami r14-ben es r15-ban levo ertekeket az r13 segedregiszter segitsegevel osszeadja
123  C 50    FE92        mov     (r2), r14
124  C 51    A740        cmp     r7, #HALF[40]           ;ha a memoria felenel vagyunk, vegeztunk
125  C 52    B655        jnn     inner_return[55]
126  C 53    0701        add     r7, #0x01               ;noveljuk a szamlalot
127  C 54    B049        jmp     inner_inner_loop[49]
128  C 55            inner_return:
```

```
129  C 55    BA00        rts
130
131  C 56            addbcd:                             ;Tudjuk, hogy r14: elso operandus, r15: masodik, r13: carry
132  C 56    FACE        mov     r10, r14                ;elmentjuk 1x
133  C 57    FB6B        xor     r11, r11                ;lenullazzuk a statuszregisztert
134  C 58    F9CF        mov     r9, r15                 ;eltaroljuk r15 erteket
135  C 59            second:
136  C 59    4F0F        and     r15, #0x0F
137  C 5A    4E0F        and     r14, #0x0F              ;also negy bit
138  C 5B    FE0D        add     r14, r13                ;hozzaadjuk az elozo carryt az elso operandushoz
139  C 5C    CD00        mov     r13, #0x00              ;nem tudjuk xorolni, mert a carry flag jo, ha megmarad, de nullaznunk kell
140  C 5D    1D00        adc     r13, #0x00              ;az esetleges carryt megorizzuk
141  C 5E    FE0F        add     r14, r15                ;osszeadjuk a ket operandust
142  C 5F    1D00        adc     r13, #0x00              ;az esetleges carryt megorizzuk ismet
143  C 60            too_big:
144  C 60    AE0A        cmp     r14, #0x0A              ;ennel nagyobbegyenlo szam nem lehet BCD
145  C 61    B765        jv      skip[65]
146  C 62    0D01        add     r13, #0x01              ;egyet biztosan kell adni a carryhez
147  C 63    2E0A        sub     r14, #0x0A              ;levonunk belole tizet, hogy az egyeseket kapjuk
148  C 64    B060        jmp     too_big[60]
149  C 65            skip:                               ;itt mar r14-ben biztosan egy valid, 10 alatti BCD szam van
150  C 65    AB00        cmp     r11, #0x00              ;ha mar egyszer lefutott, akkor vegeztunk, ha ez az elso, akkor megyunk tovabb
151  C 66    B26E        jnz     ret[6E]                 ;
152  C 67    0B01        add     r11, #0x01              ;jelezzuk, hogy egyszer mar eljutottunk ide
153  C 68    FCCE        mov     r12, r14                ;az also negy bitet elmentjuk r12-ben, ez mar megvan
154  C 69    7A00        swp     r10                     ;r10 bitjeit felcsereljuk
155  C 6A    FECA        mov     r14, r10                ;ezt beletoltjuk r14-be
156  C 6B    FFC9        mov     r15, r9                 ;az elmentett r15-ot visszatoltjuk bele (r9-ben volt eddig)
157  C 6C    7F00        swp     r15                     ;es ennek is megcsereljuk a bitjeit
158  C 6D    B059        jmp     second[59]              ;ujrakezdjuk
159  C 6E            ret:
160  C 6E    7E00        swp     r14                     ;felcsereljuk r14 bitjeit, ezzel egy helyiertekkel magasabbra kerul
```

```
161  C 6F    FE5C        or      r14, r12                ;osszeeselve az egyesekkel megkapjuk a szamot magat, ezt r14-ben fogjuk ujra hasznalni
162  C 70    BA00        rts
163
```