Digitális technika 2. házi feladat
Gépi szintű programozás

Adatok
Név: Nagy Marcell
Neptun kód: G4SDSA
E-mail cím: -
Tankör: I03
Gyakorlat kurzus: G07
Évfolyam sorszám: 206
Digit kód: 1354276
Kibővített hexadecimális kód: 14385946297C60

Nyilatkozat
A feladatokat önállóan, meg nem engedett segédeszközök használata és mások közvetlen közreműködése nélkül oldottam meg.

Összefoglalás a végeredményről
A kód bináris formában felírva: 0001 0100 0011 1000 0101 1001 0100 0110 0010 1001 0111 1100 0110 0000
Az 1-es értékű bitek száma kiolvasható ebből a formából: 22, binárisan 00010110. Ez később használható ellenőrzésre. A részfeladatok eredményei tesztelve lettek az FPGA-n is, mindhárom esetben a helyes LED-ek villantak fel a programok futtatása után.
Hogy kiférjenek 1 oldalra a forráskódok és az ASM modellek is, a gráf egyes részeit összevontam. Ilyen esetben ugyan azokat az utasításokat végeznénk el, csak más bemenetekkel, amiket a 3. részfeladatnál jelöltem is. A sorrendet a nyilak típusával próbáltam jelölni. Miután egy „ciklustörzs” végére értünk, kiválasztjuk a következő típust (ennek a sorrendjét feltűntettem), és azt az utat követjük végig. Sajnos ez rontja az áttekinthetőséget. Mindhárom részfeladatnál csak az inicializálás 3 utasítása Moore típusú, így nem jelöltem külön a dobozok kerekítettségét.

A részfeladatok algoritmusainak hatékonysága

	Utasítások száma	Adatmemória foglalás	Végrehajtási idő	Program költség
Hagyományos	13	0	205	5330
Táblázatos	20	16	115	6440
Aritmetikai	28	0	172	9632

Vélemény
A házi feladatot hatékony módszernek találom az assembly ismeretének elsajátításához, valamint a processzor utasításkészletének megismeréséhez. A feladatkiírás néhol nem volt túl egyértelmű, de a segédlet tartalmazott kiegészítéseket. Az eredmények is mutatják, néha a legegyszerűbb megoldás a leghatékonyabb. A 3. részfeladatot volt a legnehezebb megérteni, elsőnek rossz programot készítettem el.

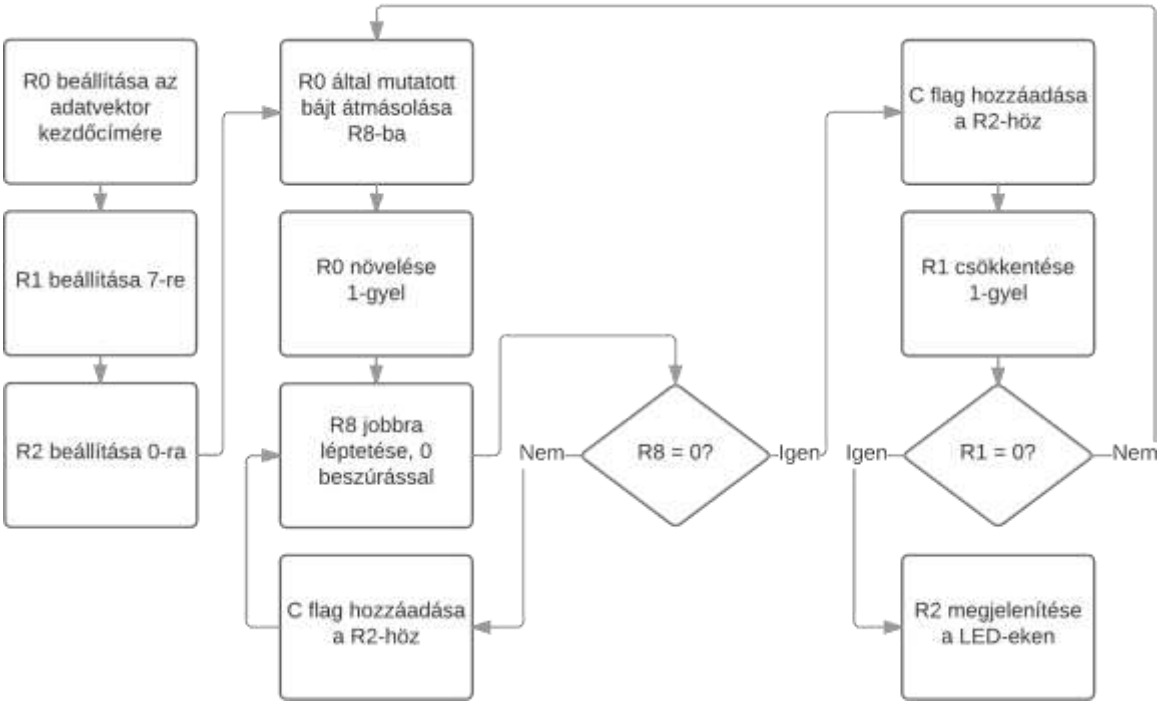
1. feladat: Hagyományos algoritmus

Ezt az algoritmust a legegyszerűbb megtervezni, amivel a tervezés közben időt lehet spórolni. Ha viszont sok adatot kell feldolgozni, sokáig tarthat a program végére érni. A bitszámlálóhoz való adás ugyan áthelyezhető lenne az ugrás elé, az algoritmust azért így építettem fel, hogy egy kis hatékonyságjavítás elérhető legyen.

A program forráskódja:

```
          DEF LD 0x80          ;A LED-ek címe
            DATA
D 00          DIGIT_CODE:     ;Átalakított, kibővített DIGIT kód
D 00            DB 0x14, 0x38, 0x59, 0x46, 0x29, 0x7C, 0x60
          CODE
C 00 C000   mov  r0, #DIGIT_CODE[00] ;Az adatvektor mutató
                                ;beállítása alaphelyzetbe
C 01 C107   mov  r1, #0x07      ;Hátralévő bájtok száma, kezdetben
                                ;adatvektor elemeinek számával (7) egyenlő
C 02 C200   mov  r2, #0x00      ;A számláló kezdeti (0) értékének megadása
C 03      read:
C 03 F8D0   mov  r8, (r0)       ;r0 által mutatott bájt átmásolása r8-ba
C 04 0001   add  r0, #0x01      ;A pointer beállítása a következő elemre
C 05      bit_loop:
C 05 F871   sr0  r8             ;A regiszter léptetése jobbra 0 beszúrással,
                                ;C értéke a 8. bit, Z értéke 1, ha a
                                ;művelet után az r8 értéke 0
C 06 B109   jz   zero[09]       ;Ha Z értéke 0, ugrunk zero-ra. Különben...
C 07 1200   adc  r2, #0x00      ;C hozzáadása a bitszámlálóhoz
C 08 B005   jmp  bit_loop[05] ;Van még 1-es bit, újrakezdjük
C 09      zero:
C 09 1200   adc  r2, #0x00      ;C hozzáadása a bitszámlálóhoz
C 0A 2101   sub  r1, #0x01      ;Adatelem számláló csökkentése 1-gyel
C 0B B203   jnz  read[03]       ;Ha r1 értéke nem 0, van még elem, amit
                                ;fel kell dolgoznunk, ugrás read-re
C 0C 9280   mov  LD[80], r2     ;Mindent feldolgoztunk, eredmény kijelzése
```

A program ASM modellje:



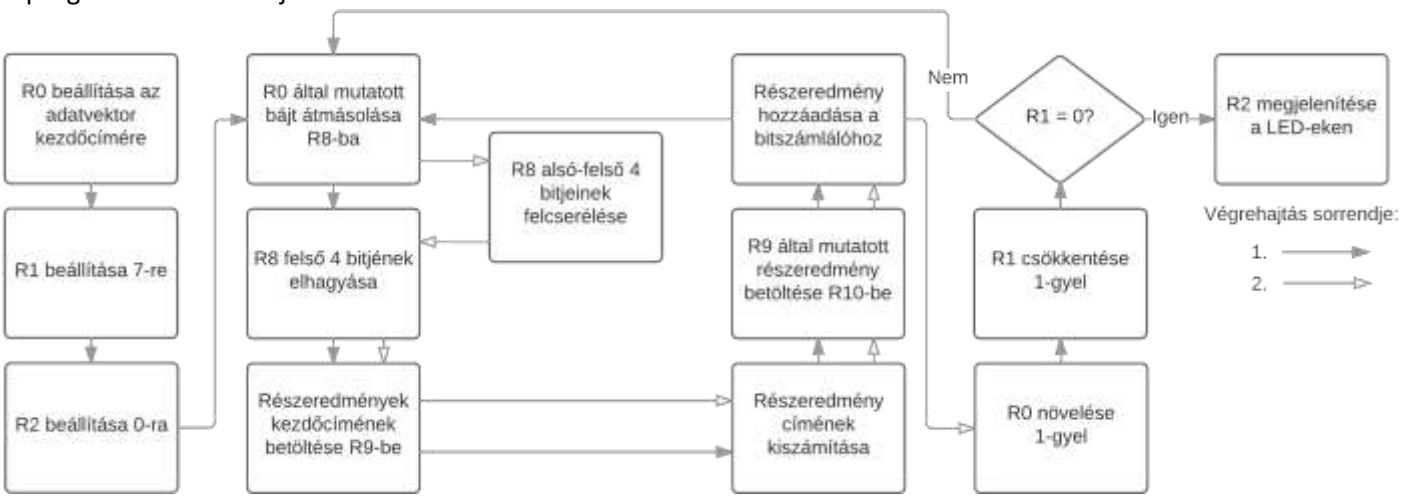
2. feladat: Táblázatos algoritmus

Hatékony módszer, ha számtalan ismétlődő részt kell feldolgoznunk, mert elegendő egyszer a részeredményeket előre kiszámolni. A DIGIT kód nem elég hosszú ahhoz, hogy érződjön az algoritmus hatása. Ugyan itt kedvező volt a 16-féle részmegoldás, más feladatnál nem biztos, hogy mindent fel lehet sorolni, és eltárolni a memóriában, vagy ha fel is lehet részekre bontani az adatokat, hogy azoknak a részeredményét keressük, az átalakítás költséges lehet.

A program forráskódja:

```
          DEF LD 0x80              ;A LED-ek címe
            DATA
D 00      DIGIT_CODE:              ;Átalakított, kibővített DIGIT kód
D 00        DB 0x14, 0x38, 0x59, 0x46, 0x29, 0x7C, 0x60
            ORG 0x10               ;Memóriacímző növelése
D 10      SUM_LUT:                 ;Eészeredmények táblázata
D 10        DB 0x00, 0x01, 0x01, 0x02, 0x01, 0x02, 0x02, 0x03
D 18        DB 0x01, 0x02, 0x02, 0x03, 0x02, 0x03, 0x03, 0x04
          CODE
C 00 C000   mov  r0, #DIGIT_CODE[00] ;Az adatvektor mutató
                                     ;beállítása alaphelyzetbe
C 01 C107   mov  r1, #0x07           ;Hátralévő bájtok száma, kezdetben
                                     ;adatvektor elemeinek számával egyenlő
C 02 C200   mov  r2, #0x00           ;A számláló kezdeti értékének megadása
C 03      read:
C 03 F8D0   mov  r8, (r0)            ;r0 által mutatott bájt átmásolása r8-ba
C 04 480F   and  r8, #0x0F           ;r8 maszkolása, alsó 4 bitet tartjuk meg
C 05 C910   mov  r9, #SUM_LUT[10]    ;A részeredmények kezdőcímének betöltése
C 06 F908   add  r9, r8              ;A részeredmény címének kiszámítása
                                     ;kezdőcím+szám helyen van a számhoz
                                     ;tartozó részeredmény a memóriában
C 07 FAD9   movv r10, (r9)           ;Részeredmény átmásolása a regiszterbe
C 08 F20A   add  r2, r10             ;Részeredmény hozzáadása a bitszámlálóhoz
C 09 F8D0   mov  r8, (r0)            ;r0 által mutatott bájt átmásolása r8-ba
C 0A 7800   swp  r8                  ;Alsó-felső 4 bit felcserélése
C 0B 480F   and  r8, #0x0F           ;r8 maszkolása, alsó 4 bitet tartjuk meg
C 0C C910   mov  r9, #SUM_LUT[10]    ;A részeredmények kezdőcímének betöltése
C 0D F908   add  r9, r8              ;A részeredmény címének kiszámítása
C 0E FAD9   mov  r10, (r9)           ;Részeredmény átmásolása a regiszterbe
C 0F F20A   add  r2, r10             ;Részeredmény hozzáadása a bitszámlálóhoz
C 10 0001   add  r0, #0x01           ;A pointer beállítása a következő elemre
C 11 2101   sub  r1, #0x01           ;Adatelem számláló csökkentése 1-gyel
C 12 B203   jnz  read[03]            ;Ha r1 értéke nem 0, van még elem, amit
                                     ;fel kell dolgoznunk, ugrás read-re
C 13 9280   mov  LD[80], r2          ;Minden feldolgozva, eredmény kijelzése
```

A program ASM modellje:



3. feladat: Aritmetikai algoritmus

A MiniRISC processzor utasításkészletében nincs olyan parancs, amivel egyszerre több bittel is lehetne léptetni a bájtot, a sok parancs használata látszik a költségen is. Nagyobb méretű adatokon éri meg használni.

A program forráskódja:

```
            DEF LD 0x80         ;A LED-ek címe
            DATA
D 00          DIGIT_CODE:       ;Átalakított, kibővített DIGIT kód
D 00            DB 0x14, 0x38, 0x59, 0x46, 0x29, 0x7C, 0x60
            CODE
C 00 C000     mov  r0, #DIGIT_CODE[00] ;Az adatvektor mutató
                                ;beállítása alaphelyzetbe
C 01 C107     mov  r1, #0x07    ;Hátralévő bájtok száma, kezdetben adatvektor
                                ;elemeinek számával (7) egyenlő
C 02 C200     mov  r2, #0x00    ;A számláló kezdeti (0) értékének megadása
C 03        read:
C 03 F8D0     mov  r8, (r0)     ;r0 által mutatott bájt átmásolása r8-ba
C 04 0001     add  r0, #0x01    ;A pointer beállítása a következő elemre
C 05 F9C8     mov  r9, r8       ;Bájt lemásolása r9-be
C 06 4855     and  r8, #0x55    ;Páratlan bitek maszkolása
C 07 49AA     and  r9, #0xAA    ;Páros bitek maszkolása
C 08 F971     sr0  r9           ;Páros bitek jobbra léptetése, 0 beszúrással
C 09 F809     add  r8, r9       ;A 4-4 érték összeadása, az eredmény a bájt
                                ;páros-páratlan bitjei Co-S-nek felelnek meg
C 0A F9C8     mov  r9, r8       ;Részeredmények lemásolása r9-be
C 0B 4833     and  r8, #0x33    ;Páratlan számú részösszegek maszkolása,
                                ;hogy 2-es csoportokat kapjunk
C 0C 49CC     and  r9, #0xCC    ;Páros számú részösszegek maszkolása
C 0D F971     sr0  r9           ;r9 jobbra léptetése, 0 beszúrással, először...
C 0E F971     sr0  r9           ;Majd másodszor (mivel 2-es csoportok vannak)
C 0F F809     add  r8, r9       ;A 2-2 érték összeadása
C 10 F9C8     mov  r9, r8       ;Részeredmények lemásolása r9-be
C 11 4807     and  r8, #0x07    ;Páratlan számú részösszegek maszkolása
C 12 4970     and  r9, #0x70    ;Páros számú részösszegek maszkolása
C 13 F971     sr0  r9           ;r9 jobbra léptetése, beszúrással, először...
C 14 F971     sr0  r9           ;Majd másodszor...
C 15 F971     sr0  r9           ;Harmadszor...
C 16 F971     sr0  r9           ;És negyedszer (mivel 3-as csoportok vannak
                                ;+1 nem használt bit (a 4.))
C 17 F809     add  r8, r9       ;A két érték teljes összeadása
C 18 F208     add  r2, r8       ;Részeredmény hozzáadása a bitszámlálóhoz
C 19 2101     sub  r1, #0x01    ;Adatelem számláló csökkentése 1-gyel
C 1A B203     jnz  read[03]     ;Ha r1 nem nulla, ugrás read-re
C 1B 9280     mov  LD[80], r2 ;Mindent feldolgoztunk, eredmény kijelzése
```

A program ASM modellje: