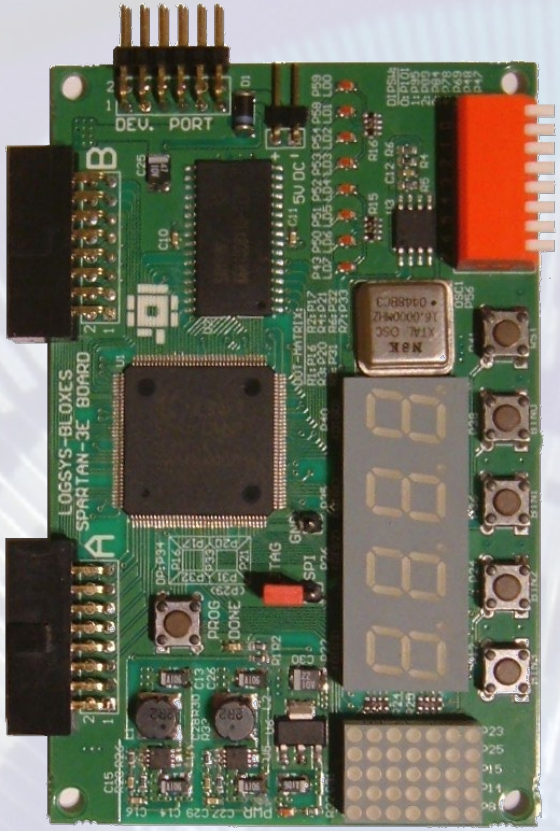
BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

Digitális technika (VIMIAA02)
8. laboratórium

Raikovich Tamás
BME MIT

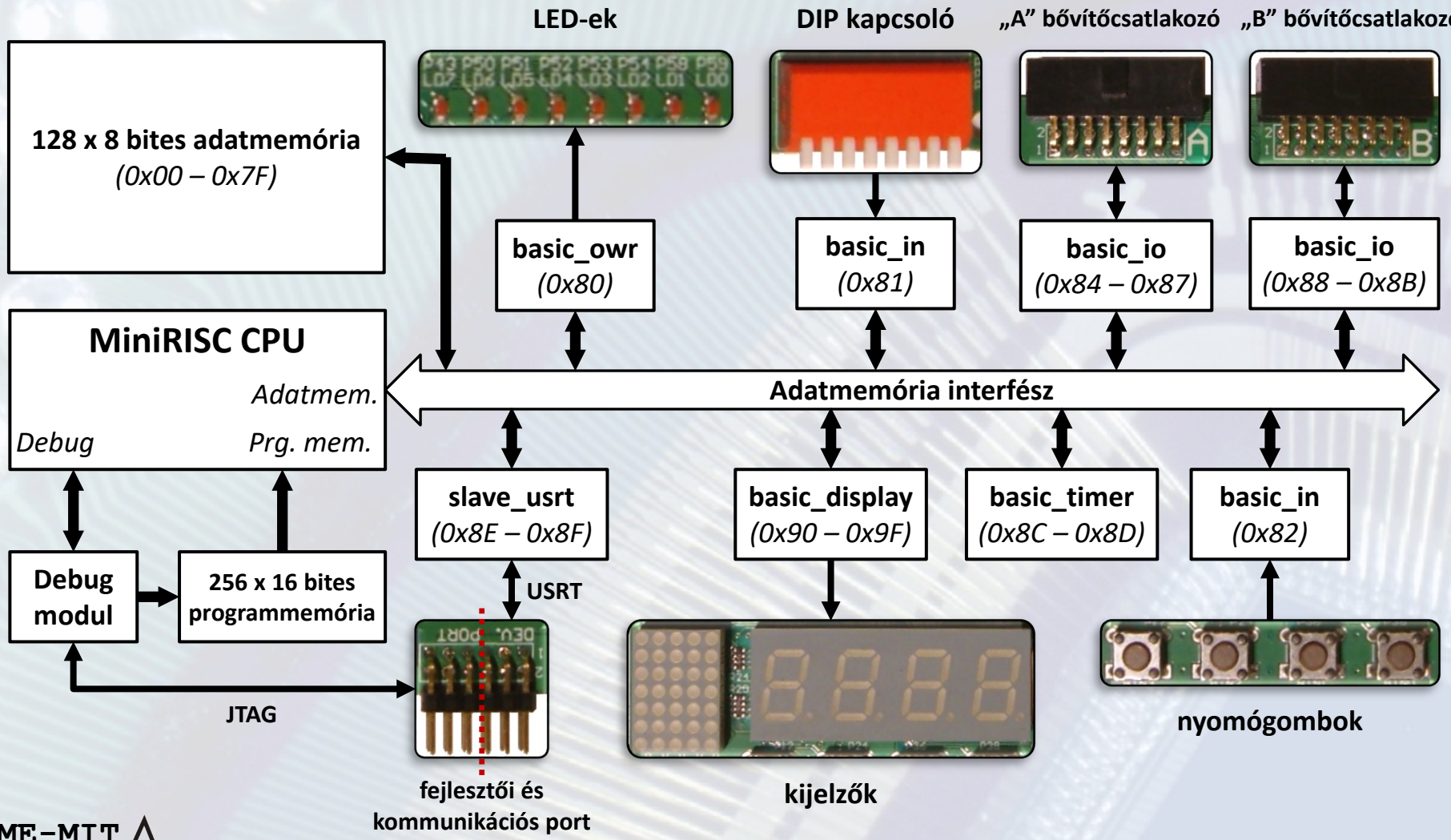
Bevezetés: MiniRISC processzor

- **8 bites vezérlőegység egyszerű alkalmazásokhoz**
- **Jól illeszkedik a LOGSYS Spartan-3E FPGA kártya komplexitásához**
- **Egyszerű felépítés, kis erőforrásigény**
- **Harvard architektúra**
  - 256 x 16 bites programmemória
  - 256 x 8 bites adatmemória
- **Egyszerű RISC jellegű utasításkészlet**
  - Load/store architektúra
  - Műveletvégzés csak regisztereken
  - 16 x 8 bites belső regisztertömb



Bevezetés: MiniRISC mintarendszer

(Egyszerűsített Blokkvázlat)



Bevezetés: MiniRISC assembler

- **A LOGSYS Spartan-3E FPGA kártyához készült MiniRISC mintarendszerre történő programfejlesztéshez egy grafikus fejlesztői környezet áll rendelkezésre (MiniRISC IDE)**
- **Az assembly nyelven megírt programok lefordítása a LOGSYS MiniRISC assemblerrel lehetséges**
- **Az assembler a processzorhoz illesztett képességű**
  - Egyszerű utasítás értelmezés
  - Abszolút kódot generál (nincs linker)
  - Nincs makrózás
  - Nincs cím aritmetika
  - Nincs feltételes fordítás
  - Hiba esetén az első hibánál hibaüzenettel leáll

Bevezetés: MiniRISC assembler

- **Forrásfájl: egyszerű szövegfájl .s kiterjesztéssel**
- **A feldolgozás soronként történik: 1 sorban 1 utasítás**
- **Az assembly forráskód sorok formátuma**
  LABEL: MNEMONIC OP1{, OP2} ; Comment
  - ***LABEL*** Azonosító, értéke az azt követő utasítás címe
  - ***MNEMONIC*** A végrehajtandó műveletre utaló mnemonik, pl. ***ADD*** – összeadás, ***JMP*** – ugrás, stb.
  - ***OP1{, OP2}*** A művelet operandusai, ***OP2*** nincs mindig (***OP1*** sincs az ***RTS***, ***RTI***, ***STI*** és ***CLI*** utasítások esetén)
  - ***; Comment*** A ';' karakter a megjegyzés kezdete, melyet az assembler figyelmen kívül hagy
- **Javaslat**
  - Minden utasításhoz írjunk megjegyzéseket
  - A formázáshoz használjuk a TAB karaktert

Bevezetés: MiniRISC assembler

- **Kevés szabály van**
- **Az utasítások operandusai lehetnek**
  - Regiszter r0 – r15
  - Konstans #0 – #255 (ALU műveletekhez)
  - Memóriacím 0 – 255 (ez is konstans, csak más célra)
  - Indirekt cím (r0) – (r15)
- **Numerikus konstans**
  - Nincs prefix decimális 0 – 255
  - 0x prefix hexadecimális 0x00 – 0xFF
  - 0b prefix bináris 0b00000000 – 0b11111111
- **Karakter konstans**
  - A ' ' jelek közötti karakter (pl. **#'A'** – értéke 65)
  - Escape szekvenciák: **'\'', '\"', '\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v'**
- **String konstans**
  - A " " jelek közötti karakterfüzér (pl. **"MiniRISC\r\n"**)
  - Csak az adat szekcióban használható

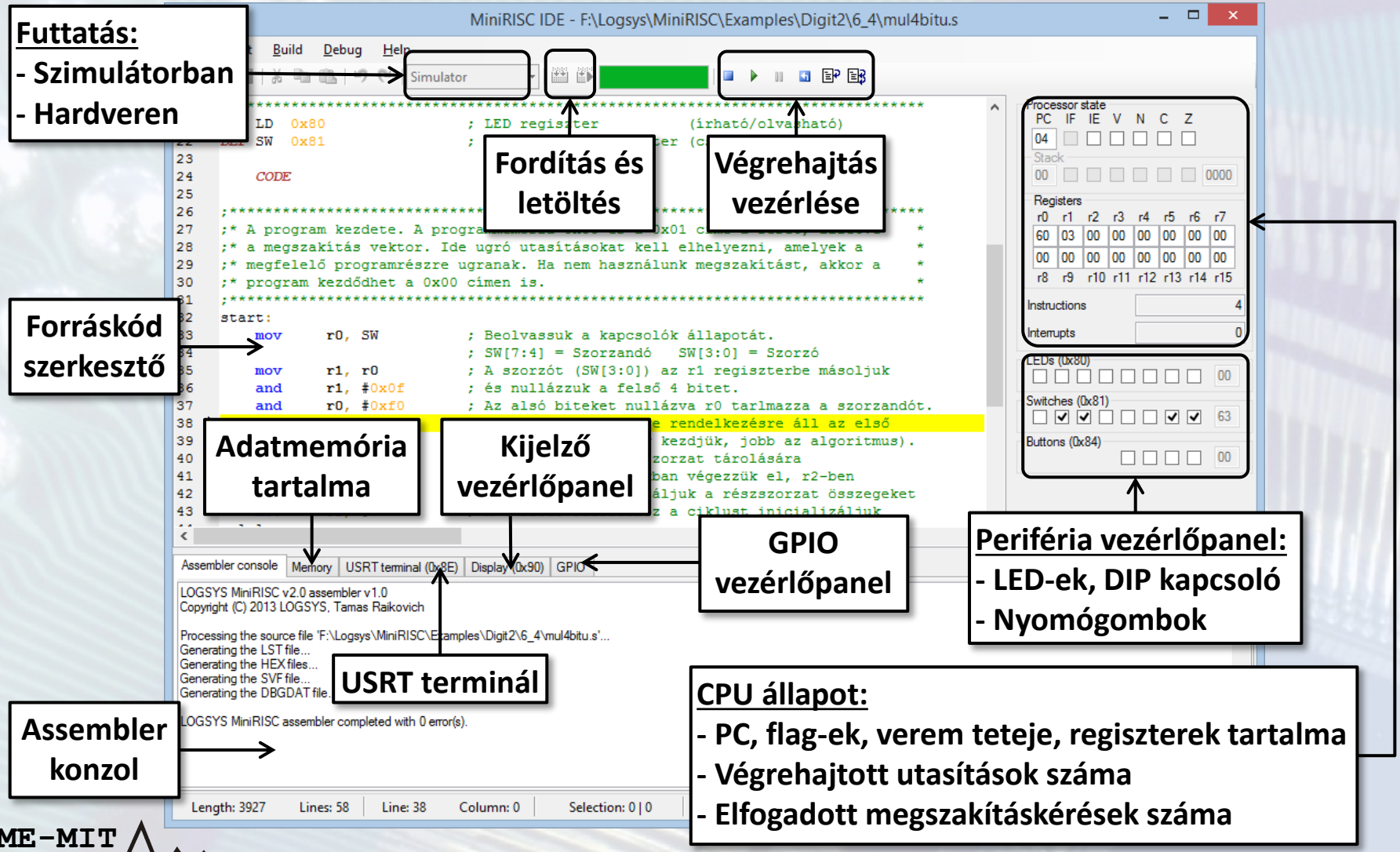
Bevezetés: MiniRISC assembler

- **Assembler direktívák**
  - ***DEF:*** azonosítót rendel konstanshoz
    DEF SW 0x81 ;A DIP kapcsoló periféria címe
    - Ez nem CPU utasítás, hanem az assembler számára definiál egy helyettesítési szabályt, ami a felhasználó számára biztosítja a forráskód jobb olvashatóságát
  - ***CODE:*** a kód szekció kezdetét jelzi
    - Az utána lévő forráskód részből generált kód a prg. memóriába kerül
  - ***DATA:*** az adat szekció kezdetét jelzi
    - Az utána lévő forráskód részből generált kód az adatmemóriába kerül
    - Az adat szekcióban csak címkék és DB direktívák lehetnek
  - ***DB:*** az adatmemória inicializálása konstansokkal
    DB "MiniRISC.\r\n", 0 ;Nulla végű string
    - Csak az adat szekcióban lehet használni
    - Numerikus, karakter és string konstansok adhatók meg utána
    - Az egyes konstansokat vesszővel kell elválasztani egymástól

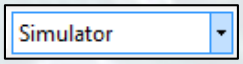
Bevezetés: MiniRISC assembler

- **Assembler direktívák**
  - ***ORG:*** kezdőcím közvetlen előírása
    ORG memóriacím
    - Az utána lévő kódrész kezdőcímét állítja be
    - Használható a kód és az adat szekcióban is
- **Az assembler által generált LST fájlban ellenőrizhető az utasítások címe és gépi kódja, valamint a fordító által használt azonosítók értelmezése**
- **Ha a kód hibátlanul lefordult, akkor beépíthető a hardver tervbe, mint memória inicializáló adat (.HEX kimeneti fájlok) vagy letölthető a kész konfigurációra (.SVF kimeneti fájl)**

Bevezetés: MiniRISC IDE



Bevezetés: MiniRISC IDE

- **A** 

legördülő menüből kiválasztható, hogy a program
  - A szimulátorban fusson (ez mindig rendelkezésre áll)
  - A csatlakoztatott hardveren fusson (***LDCxxx*** letöltőkábel)
  - Választás csak akkor lehetséges, ha program nem fut
- **Szimulátor**
  - Órajelciklus pontossággal szimulálja az FPGA áramkörben megvalósított processzoros rendszert
  - A program végrehajtása itt lassabb, mint a hardveren
    - Kb. 400000 utasítás/s → **kb. 1,2 MHz rendszerórajel frekvencia**
  - A MiniRISC mintarendszerben lévő perifériák nagy része a szimulátorban is elérhető
- **Futtatás hardveren**
  - Ha van FPGA kártya csatlakoztatva a számítógéphez

Bevezetés: MiniRISC IDE

- **A program a *Compile*** ( , **F5) gombbal fordítható le**
  - A hibaüzenetek a konzolban jelennek meg
  - A hibás programsorok pirossal aláhúzásra kerülnek
- **A lefordult program a *Download*** ( , **F6) gombbal tölthető le**
  - Hardveren történő futtatás esetén ha még nincs felkonfigurálva az FPGA, akkor először ez történik meg
  - A program futása megáll a 0x00 címen (reset vektor)
- **A program végrehajtásának vezérlése**
  - ***Run*** ( , F7): a program végrehajtásának folytatása
  - ***Break*** ( , F8): a program futásának felfüggesztése, a következő végrehajtandó utasítást sárga kiemelés jelzi
  - ***Step*** ( , F10): az aktuális utasítás végrehajtása, a program futása megáll a következő utasításnál

Bevezetés: MiniRISC IDE

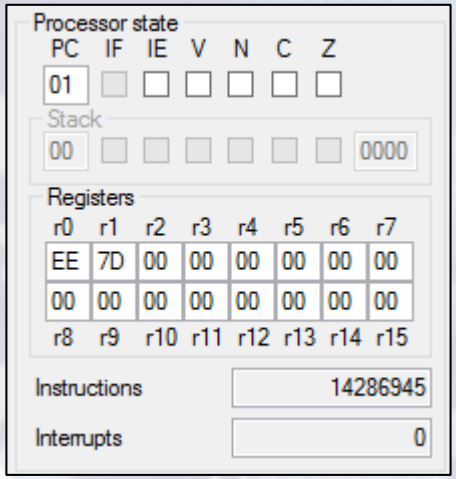
- **A program végrehajtásának vezérlése**
  - ***Auto step*** (): a ***Step*** parancs automatikus kiadása
    - A gyakoriság a Debug menüben állítható be
    - Az automatikus léptetés a ***Break*** paranccsal állítható le
  - ***Reset*** ( , F9): reset jel kiadása a processzoros rendszernek
  - ***Stop*** (): a program futtatásának befejezése
    - Ezután ismételt programletöltés szükséges
- **Töréspont elhelyezése a szerkesztőben a margóra történő kattintással lehetséges**
  - A töréspontot piros kör jelzi a margón

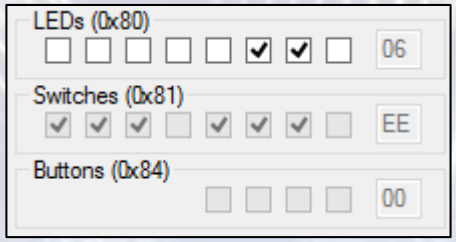
```
10        mov     r0, SW      ; A kapcsolók állapota r0-ba kerül.
11        mov     LD, r0      ; A LED-ekre kiírjuk r0 értékét.
12        jmp     start       ; Ugrás a ciklus elejére.
```

  - Töréspontra futáskor a program végrehajtása megáll
    - Lényegében egy ***Break*** parancs kiadása történik meg hardveresen

Bevezetés: MiniRISC IDE

- **A processzor állapotának és az adatmemória tartalmának módosítása, illetve a perifériák vezérlése csak felfüggesztett programvégrehajtás (break állapot) esetén lehetséges**
- ***Processor state* panel:** a processzor állapota
  - Programszámláló (PC) értéke
  - Flag-ek (Z, C N, V, IE, IF) értéke (IF csak olv.)
  - Verem tetejének értéke (csak olvasható)
  - Regiszterek értéke
  - A végrehajtott utasítások száma
  - Az elfogadott megszakításkérések száma
- **Alap perifériák vezérlőpanelje**
  - A LED-ek, a kapcsolók és a nyomógombok állapotát jeleníti meg
  - Biztosítja a perifériák regiszterszintű vezérlését





A programfejlesztés lépései

- Gondoljuk át a problémát és készítsünk vázlatos elképzelést, hogy mit és hogyan szeretnénk megoldani
- Fogalmazzuk meg a megoldást a MiniRISC assembly utasítások szintjén és készítsük el a forráskódot
- A begépelt programot fordítsuk le a Compile (F5) paranccsal
- Az esetleges hibákat javítsuk ki
- A hibátlanul lefordult programból generált SVF fájl letölthető a Download (F6) paranccsal
- Ellenőrizzük a program működését debug módban a MiniRISC IDE szolgáltatásainak segítségével

Bevezető feladatok

- **A MiniRISC processzor programozásának gyakorlásához a félév első laborjának egyszerű Verilog HDL alapú bevezető feladatait ismételjük meg gépi szintű programozással**
- **A feladatok így ismertek, csak egy kicsit más szemlélet szükséges a realizációhoz**
- **Sok esetben az új megközelítés sokkal egyszerűbb, sok esetben itt ütközünk nehézségekbe**
- **Nézzük át a MiniRISC processzor utasításkészletét!**
  - Elérhető a MiniRISC IDE Help menüjében
- **Sok sikert a gépi szintű programozáshoz!**

1. feladat: assembly „Hello World!”

- **Folyamatosan jelenítsük meg a kapcsolók állapotát a LED-eken**
  - Hogy tudjuk elérni a perifériákat?
  - Milyen utasítások szükségesek ehhez?
- **Rajzoljuk fel a program folyamatábráját**
- **Készítsük el a programot a MiniRISC IDE-ben**
- **Próbáljuk ki a program működését az FPGA kártyán**

2. feladat: kettes komplemens képzés

- **Folyamatosan jelenítsük meg a kapcsolón beállított érték kettes komplemensét a LED-eken**
  - Hogy tudunk kettes komplemenst képezni?
  - Milyen utasítások szükségesek ehhez?
- **Rajzoljuk fel a program folyamatábráját**
- **Egészítsük ki az előző feladatban elkészített kódot**
- **Próbáljuk ki a program működését az FPGA kártyán**

3. feladat: összeadó

- **Folyamatosan jelenítsük meg a kapcsolón beállított érték alsó és felső 4 bitjének összegét a LED-eken**
  - SW[3:0] + SW[7:4] → LD
  - Az operandusok előjel nélküli értékek
  - Hogy kell egymáshoz igazítani az operandusokat?
  - Milyen utasítások szükségesek ehhez?
- **Készítsük el a programot a MiniRISC IDE-ben**
- **Próbáljuk ki a program működését az FPGA kártyán**
  - Mit tapasztalunk, ha az operandusokat kettes komplemens kódolásúnak értelmezzük?
  - Milyen módosítás szükséges, hogy az eredmény 8 biten előjel helyesen jelenjen meg a LED-eken?

4. feladat: szorzó

- **Folyamatosan jelenítsük meg a kapcsolón beállított érték alsó és felső 4 bitjének szorzatát a LED-eken**
  - SW[3:0] · SW[7:4] → LD
  - Az operandusok előjel nélküli értékek
- **A MiniRISC processzornak nincs szorzás utasítása**
  - Vegyük alapul az írásbeli szorzást
  - Elemezzük az algoritmus működését

```
     1001 · 0110 ← szorzandó, szorzó
     0000 = (1001 & 0000) << 0 ← 1. részszorzat
    10010 = (1001 & 1111) << 1 ← 2. részszorzat
   100100 = (1001 & 1111) << 2 ← 3. részszorzat
+ 0000000 = (1001 & 0000) << 3 ← 4. részszorzat
 00110110 ← szorzat
```

4. feladat: szorzó

- Rajzoljuk fel a program folyamatábráját
- Készítsük el a programot a MiniRISC IDE-ben
- Próbáljuk ki a program működését az FPGA kártyán