

Digitális technika 2 első laborgyakorlat

A foglalkozás célja:

Egy olyan mikrokontrolleres fejlesztői környezet megismerése, amely tartalmazza a szövegszerkesztőt, a fordítót, a szimulátort és valamilyen letöltő/hardver debug egységeket.

A szimulátor használatának elsajátítása a következő tevékenységek során:

- Egyszerű program lépésenkénti végrehajtása, flag-ek és regiszterek (watch) megfigyelése
- Töréspont elhelyezése
- Hiba keresése/javítása adott rutinban
- Változók, SFR-ek megjelenítése, érték módosítása
- Kód futási idejének mérése

A megismerés során először előre elkészített működő projekteket kell betölteni, majd a programok működését a szimulátorban lépésenként elemezni, a működési hibákat megtalálni.

Ezt követően egyszerű algoritmust kell készíteni, szimulátorban futtatni és dokumentálni.

Ez után egy kész programot kell letölteni a fejlesztő eszközbe, és ki kell próbálni a működését.














Végül egy egyszerű programot kell módosítani és a módosítást a fejlesztő eszközben kipróbálni.

A fejlesztői környezet kezelő felülete:

The screenshot displays the MPLAB X IDE v5.50 interface. The main window is titled "ellenorzofoadatok" and shows the "Asm Source" view. The code is written in assembly language, including comments in Hungarian. The "Program Memory" window on the right shows the disassembly of the code, with columns for Line, Address, Opcode, Label, and Disassembly. The bottom status bar indicates the cycle count as 265795 (8,306094 ms) and the instruction frequency as 32 MHz. The bottom right corner shows the status "debugger halted" and "61:1 INS".

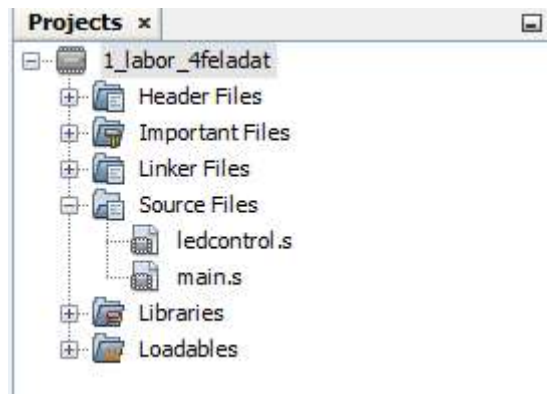
| Line | Address | Opcode | Label | Disassembly |
|------|---------|--------|----------------------|-------------|
| 248 | 001EE | EB0080 | CLR W1 | |
| 249 | 001F0 | 200202 | MOV #0x20, W2 | |
| 250 | 001F2 | 784190 | MOV.B [W0], W3 | |
| 251 | 001F4 | 40C083 | ADD.B W1, W3, W1 | |
| 252 | 001F6 | E80000 | INC W0, W0 | |
| 253 | 001F8 | E90102 | DEC W2, W2 | |
| 254 | 001FA | 3AFFFB | BRA NZ, ell_ck | |
| 255 | 001FC | 784801 | MOV.B W1, [W0] | |
| 256 | 001FE | 784001 | MOV.B W1, W0 | |
| 257 | 00200 | 060000 | RETURN | |
| 258 | 00202 | EB0080 | CLR W1 | |
| 259 | 00204 | 09001F | REPEAT #0x1F | |
| 260 | 00206 | 40C0B0 | ADD.B W1, [W0++], W. | |
| 261 | 00208 | 784801 | MOV.B W1, [W0] | |
| 262 | 0020A | 784001 | MOV.B W1, W0 | |
| 263 | 0020C | 060000 | RETURN | |
| 264 | 0020E | BE9F80 | MOV.D W0, [W15++] | |
| 265 | 00210 | BE9F82 | MOV.D W2, [W15++] | |
| 266 | 00212 | BE9F84 | MOV.D W4, [W15++] | |
| 267 | 00214 | F80042 | PUSH SR | |
| 268 | 00216 | 208000 | MOV #0x800, W0 | |
| 269 | 00218 | 209001 | MOV #0x900, W1 | |
| 270 | 0021A | 20A002 | MOV #0xA00, W2 | |
| 271 | 0021C | 20B003 | MOV #0xB00, W3 | |
| 272 | 0021E | EB0200 | CLR W4 | |
| 273 | 00220 | 7802B1 | MOV [W1++], W5 | |
| 274 | 00222 | 529832 | SUB W5, [W2++], [W0. | |
| 275 | 00224 | AE6042 | BITSS W4, #3 | |
| 276 | 00226 | E80204 | INC W4, W4 | |
| 277 | 00228 | E90183 | DEC W3, W3 | |
| 278 | 0022A | 3AFFFA | BRA NZ, tk_ck | |

Legfontosabb, a mérések során használandó funkciók:

| | Menü | Funkció |
|---|---|---|
|  | File - Open project | Projekt állomány megnyitása. A projektállományok könyvtárakban találhatók, megnyitás során a kívánt könyvtárat kell kijelölni. A könyvtárak elnevezése .X kiterjesztéssel rendelkezik |
| | File - Close project | Az aktuális projekt bezárása. A módosítások automatikusan mentésre kerülnek |
|  | Production - Clean and Buld Project | Az előző fordítások eredményeit törli, újra fordítja és linkeli a projektet. |
|  | Production - Make and Program Device | Lefordítja a programot és betölti a panelbe. Ha szimulátor van kijelölve, a funkció nem aktív. |
|  | Debug - Debug Project | Elindítja a programot. Ha szimulátor van kijelölve, a szimulátort indítja, ha hardver eszköz, letölti a programot és elindítja. Amennyiben van töréspont beállítva, az elsőnél megáll. Egyébként a program végtelenül fut. |
|  | Debug - Reset | Alaphelyzetbe állítja a programot |
|  | Debug - Step Over | Végrehajtja az aktuális utasítást és megáll a következő utasításon. Ha az utasítás szubrutinhívás, a teljes szubrutint végrehajtja. |
|  | Debug - Step Into | Végrehajtja az aktuális utasítást és megáll a következő utasításon. Ha az utasítás szubrutinhívás, a belép a szubrutinba és megáll az első utasításon. |
|  | Debug - Continue | Folytatja a program futtatását az aktuális PC értéktől. A futás a következő töréspont elérésekor áll meg. |
|  | Debug - Run to Cursor | Futtatja a programot és megállítja, ha az a kurzor által kijelölt sorra ér. Ha nem érinti a kijelölt sort, a program futása nem áll meg! |
|  | Debug - Set PC at Cursor | A programszámláló értékét a kurzor sorára állítja. A program futása innen folytatható. A változók, regiszterek értéke nem változik! |
|  | Debug - Pause | Megállítja a program futását. A kurzor az aktuális programsorra mutat |
|  | Debug - Finish Debugger Session | Befejezi a program követését. A program a Debug Project funkcióval indítható el ismét. |
|  | Run Project | Lefordítja a programot, letölti a kontrollerbe és futtatja debug lehetőség nélkül. Ha lecsatlakoztatjuk a tápfeszültséget és újra csatlakoztatjuk a kontroller mindig ezt a programot fogja futtani a következő felülírásig. A gomb csak akkor aktív, ha a fejlesztő panel csatlakoztatva van és ki is van választva. |

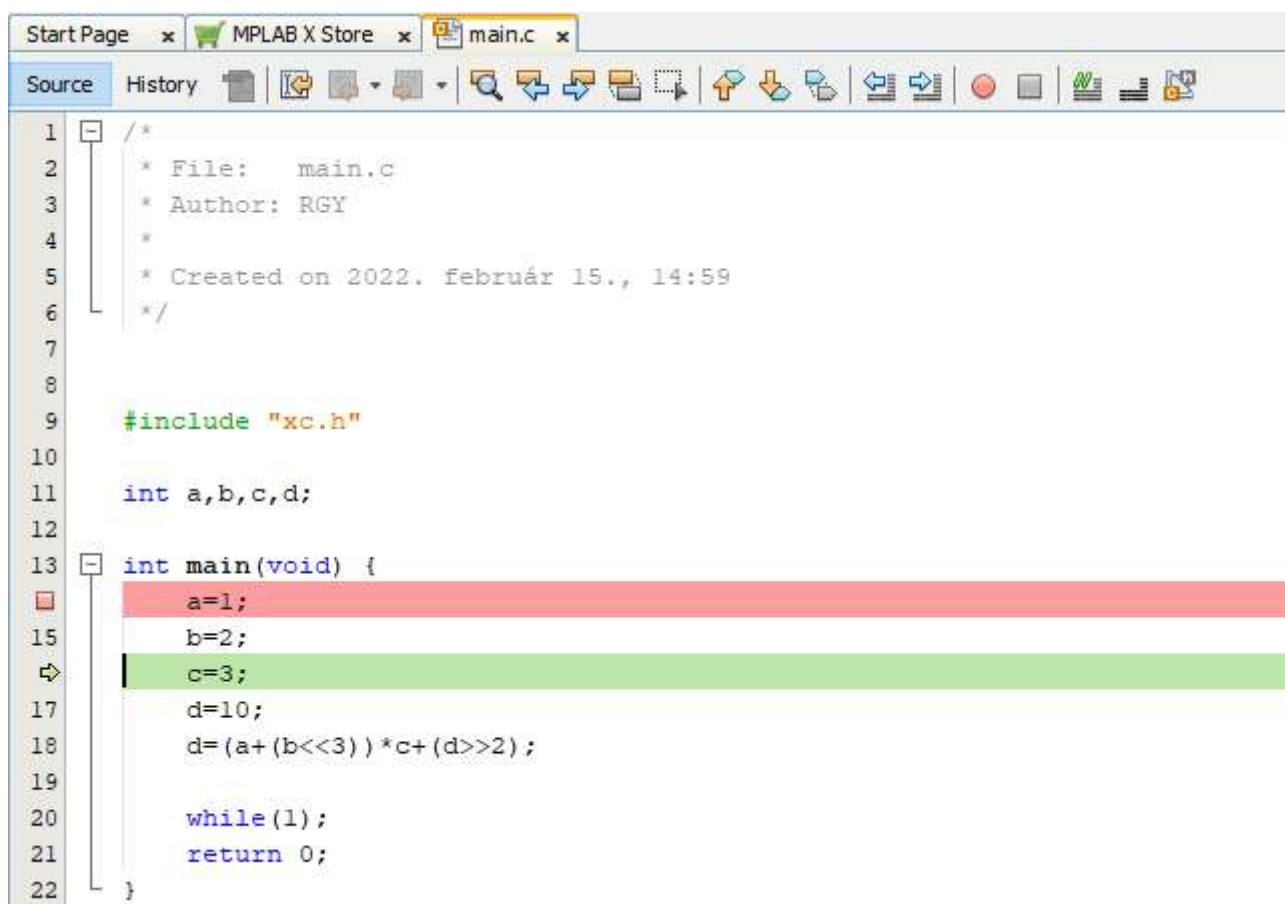
Ablakok


Window – Projects



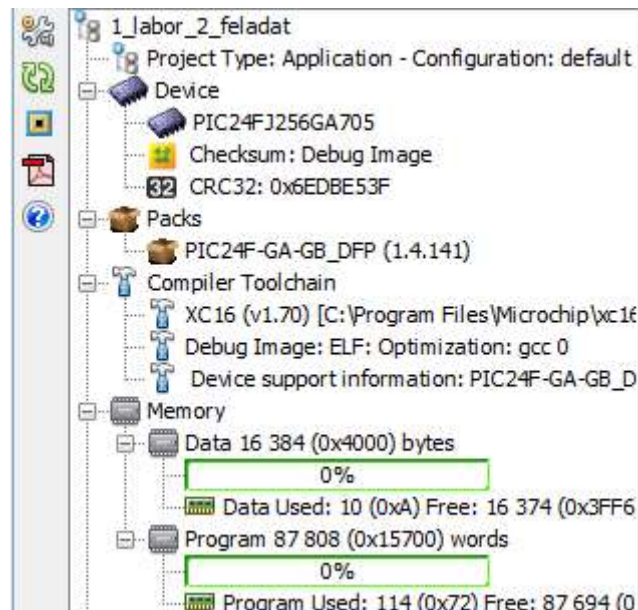
Az ablakban láthatjuk fa nézetben a projektünkhöz tartozó állományokat. A Source Files alatt a forrás állományokat találjuk. Az assembly források .s kiterjesztéssel rendelkeznek. A forrásállomány nevére duplán kattintva megnyílik a forrás ablak.

Forrás ablak

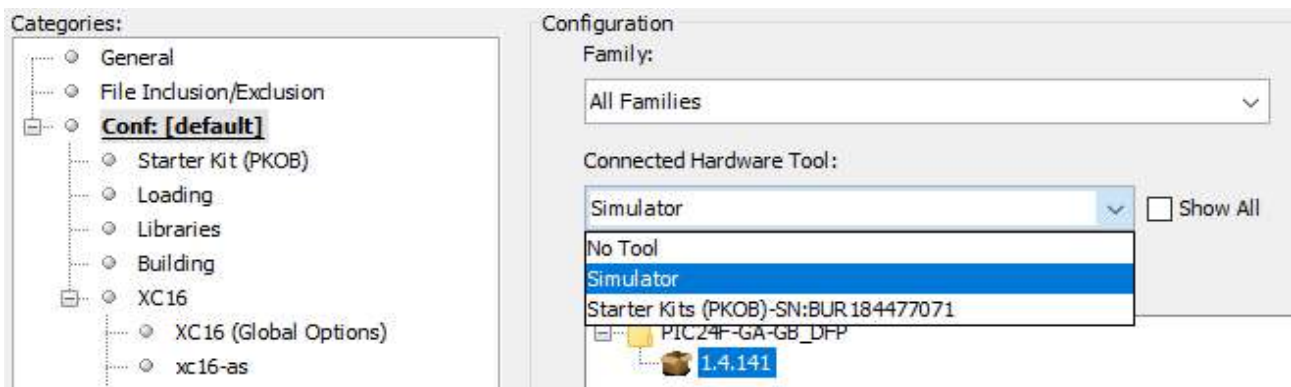


A forrás ablakban van lehetőségünk a programunkat módosítani. Nyomkövetéskor az ablak bal oldalán a programsor sorszáma kattintással töréspontot állíthatunk a program soraira. Ha a soron volt töréspont, a kattintással töröljük azt. A törésponttal rendelkező sorok piros háttérrel jelennek meg. Lépésenkénti végrehajtás során az aktuális, még éppen végre nem hajtott sor zöld háttérrel jelenik meg és a sor előtt a sorszám helyén a  szimbólum látszik.

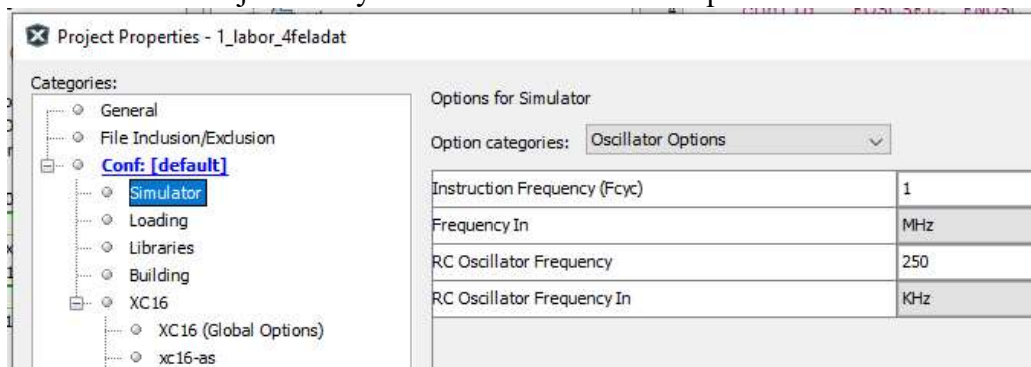
Window – Dashboard



Az ablakban a projekt beállításait látjuk. Properties funkciójával állíthatjuk be, hogy a programunkat hardver eszközön, vagy a szimulátoron szeretnénk kipróbálni.



Szimulátor esetén az aktuális órajel frekvenciát (az alapértelmezett érték 1 MHz) is itt állíthatjuk be. A mérés során használt panel órajel frekvenciája 16 MHz. Ezt az értéket kell beállítanunk, ha a szimulátor időmérő funkciójában helyes értékeket szeretnénk kapni.



Window – Debugging – Watches

| Name | Type | Address | Value |
|---|-----------------------|---------|--------|
| <input checked="" type="checkbox"/> a | (2) Bytes (User size) | 0x800 | 0x0000 |
| <input checked="" type="checkbox"/> b | (2) Bytes (User size) | 0x802 | 0x0000 |
| <input checked="" type="checkbox"/> c | (2) Bytes | 0x804 | 0x0000 |
| <input checked="" type="checkbox"/> WREG0 | SFR | 0x0 | 0x0000 |

Változók, regiszterek értékeinek figyelése. A változókra elnevezésük szerint hivatkozhatunk, a regiszterek elnevezése: w0 – WREG0, w1 – WREG1,

Az egyes elemekre a jobb gomb megnyomására felugró menüben módosíthatjuk a változók méretét

– User Defined Size: 8,16,24 ... 64 bit,

és számbábrázolási módját

– Display Value Column As: decimális, hexadecimális, bináris.

Window – Debugging – Stopwatch

| | |
|--|---|
| | Target halted. Stopwatch cycle count = 5 (312,5 ns) |
| | Target halted. Stopwatch cycle count = 6 (375 ns) |
| | Target halted. Stopwatch cycle count = 7 (437,5 ns) |
| | Stopwatch cleared. Stopwatch cycle count = 0 (0 ns) |
| | Target halted. Stopwatch cycle count = 1 (62,5 ns) |
| | Target halted. Stopwatch cycle count = 2 (125 ns) |

A program futási idejének mérése órajel ciklus és idő egységben. Az órajel frekvenciáját a Dashboard

Properties funkciójával a helyes értékre (a mérés során használt panel esetén 16Mhz) be kell állítani.

Az időmérés az ablak jobb oldalán található Clear Stopwatch nyomógommbal nullázható. Az ablak tartalma a Clear History nyomógommbal törölhető.

Window – Simulator – Stimulus

| Asynchronous Pin/Register Actions | | | | | |
|-----------------------------------|-----|----------|-------|-------|----------|
| Fire | Pin | Action | Value | Units | Comments |
| | AN0 | Set High | | | |
| | RA0 | Toggle | | | |

Ebben az ablakban lehetőség van portlábak értékét szimuláció közben megadni, megváltoztatni. A portláb neve a „Pin” legördülő menüből választható ki (pl. a B port 2. bitje = RB2), de beírással gyorsabban beállítható. Az elvégzendő műveletet az „Action” oszlopban kell beállítani.

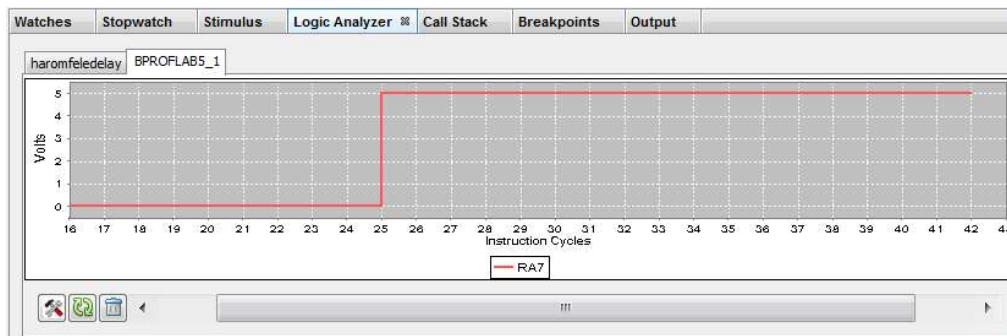
Set High – a Pin 1 értékű lesz


Set Low – a Pin 0 értékű lesz

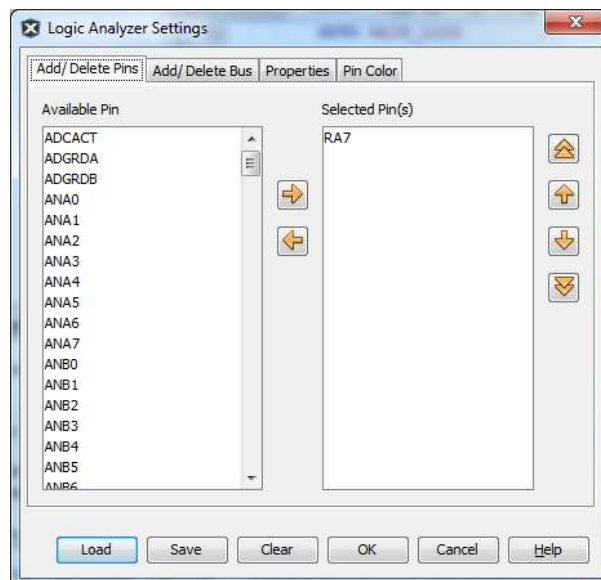
Toggle – a Pin ha 0 értékű volt 1 lesz, ha 1 értékű 0.

Az értékadást a „Fire” oszlopban elhelyezkedő gomb megnyomásával lehet végrehajtani. Fontos, hogy ha le van állítva a szimuláció, akkor az egymás után végrehajtott változtatások közül csak a legutolsó lesz hatással a programunkra. „RUN” módban viszont a lenyomás pillanatában alkalmazásra kerül az új érték.

Window – Simulator – Logic Analyzer




A bemenetek és a kimenetek értéke a Logic Analyzer ablakban idődiagramban is megjeleníthetők. A megjeleníteni kívánt jeleket a Settings  gomb megnyomása után felugró ablakban lehet megadni (Add/Delete Pins). Az egyes jelekhez ugyancsak itt lehet színeket rendelni (Pin Color)

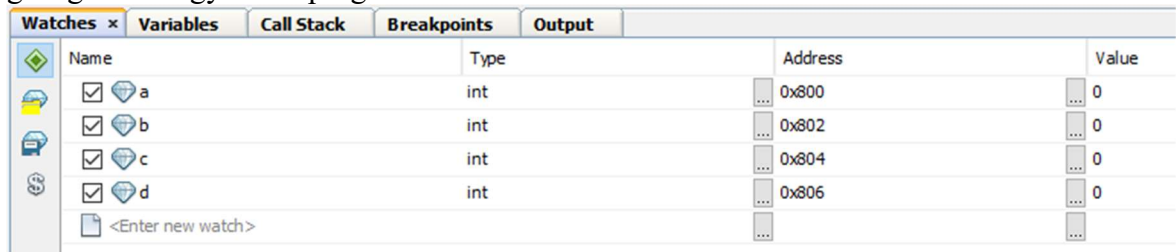


Szimulátor kipróbálása


Töltse le a Villamoskari oktatási portálról (edu.vik.bme.hu) az „1_labor_0_feladat.X.zip” projektet, majd tömörítse ki az asztalra. Indítsa el az MPLAB X IDE v5.50 fejlesztői környezetet, majd „File→Open Project...” funkcióval töltse be ezt a projektet.

Feladatok:

0. Nyissa meg a forrásfájlok közül a „main.c” állományt.
1. Számolja ki papíron, hogy mennyi lesz a „d” változó értéke a kód lefutása után.
2. Tegyen töréspontot a „main” függvény első műveletére ($a = 1$).
3. Indítsa el a szimulátort a  gombbal.
4. Hozzon létre egy saját „Watch window” ablakot a „Window→Debugging→Watches” segítségével. Vegye fel a program változóit.



| Watches × | Variables | Call Stack | Breakpoints | Output | |
|-------------------------------------|-------------------|------------|-------------|--------|--|
| | Name | Type | Address | Value | |
| <input checked="" type="checkbox"/> | a | int | 0x800 | 0 | |
| <input checked="" type="checkbox"/> | b | int | 0x802 | 0 | |
| <input checked="" type="checkbox"/> | c | int | 0x804 | 0 | |
| <input checked="" type="checkbox"/> | d | int | 0x806 | 0 | |
| | <Enter new watch> | | | | |

5. Az F7 funkciógomb segítségével lépésenként figyelje meg a program működését, a változók értékeinek alakulását.
6. A program lefutását követően ellenőrizze, hogy a 2. pontban meghatározott értéket kaptuk-e.
7. Resetelje a programot a  gomb segítségével (vagy a „Debug→Reset” menüponttal).
8. Nyissa meg a „Window→Debugging→Disassembly” ablakot.
9. Figyelje meg, az assembly kódot, a program által használt processzor regisztereket adja hozzá a Watch ablakhoz.
10. Az F7 funkciógomb segítségével lépésenként végrehajtva figyelje meg a program működését.
11. Zárja be a projektet a „File→Close All Projects” funkcióval.

A feladathoz használt C nyelvű kód:

```
#include "xc.h"

int a,b,c,d;

int main(void) {
    a = 1;
    b = 2;
    c = 3;
    d = 10;

    d=(a+(b<<3))*c+(d>>2);

    while(1);
    return 0;
}

!int main(void) {
0x20A: LNK #0x0
!    a = 1;
0x20C: MOV #0x1, W0
0x20E: MOV W0, a
!    b = 2;
0x210: MOV #0x2, W0
0x212: MOV W0, b
!    c = 3;
0x214: MOV #0x3, W0
0x216: MOV W0, c
!    d = 10;
0x218: MOV #0xA, W0
0x21A: MOV W0, d
!
!    d=(a+(b<<3))*c+(d>>2);
0x21C: MOV b, W0
0x21E: SL W0, #3, W1
0x220: MOV a, W0
0x222: ADD W1, W0, W1
0x224: MOV c, W0
0x226: MUL.SS W1, W0, W0
0x228: MOV W0, W1
0x22A: MOV d, W0
0x22C: ASR W0, #2, W0
0x22E: ADD W1, W0, W0
0x230: MOV W0, d
!
!    while(1);
0x232: BRA .L2
```


1. Hiba keresés

Adott az alábbi kód

```
.include "p24FJ256GA705.inc" ;Processzor típus kiválasztása

;config bitek
#pragma config __FOSCSEL, FNOSC_FRCPLL & PLLMODE_PLL4X & IESO_OFF
#pragma config __FOSC, POSCMD_NONE & OSCIOFCN_ON & SOSCSEL_OFF & PLLSS_PLL_FRC & IOL1WAY_OFF
#pragma config __FWDTC, FWDTCN_OFF
#pragma config __FICD, ICS_PG2 & JTAGEN_OFF

.global __reset ;kötelezően exportálandó címke, ettől kerül a reset vektor helyére a kódunk és nem lesz
semmi más a memóriában


.bss
a: .space 2
b: .space 2
c: .space 2
d: .space 2

.text
__reset:
;kötelező startup kód, ha stack hivatkozás történik ezelőtt, az reset
mov # __SP_init,w15 ; Stack pointer inicializálás
mov # __SPLIM_init,w0 ;
mov w0,__SPLIM ; stack limit inicializálás


main:
;feladat: találjuk meg mi a hiba az alábbi kódban:
;A következő kódrészlet 3 előjel nélküli egész változóból számol valamit
;írjuk fel a képletet, próbáljuk ki ha pl a=100, b=200, c=5, 12-t kellene adjon, vajon miért nem annyit ad?

mov b, W0
mov a, W1
add W1, W0, W0
sl W0, 2, W1
mov 800, W0
sub W1, W0, W0
mov W0, W1
mov c, W0
asr W1, W0, W0
mov W0, d

vegtelen:
bra vegtelen
.end
```

1. Töltse le a Villamoskari oktatási portálról (edu.vik.bme.hu) az „1_labor_1_feladat.X.zip” projektet, majd tömörítse ki az asztalra. Indítsa el az MPLAB X IDE v5.50 fejlesztői környezetet, majd „File→Open Project...” funkcióval töltse be ezt a projektet.
2. Nyissa meg a „main.s” forráskódot.
3. Fordítsa le a kódot a  (Clean and build project) funkcióval. Értelmezze a kapott hibaüzenetet és javítsa meg a hibát.

4. A készítőtől kapott információk szerint (a=100, b=200, c=5) esetén az eredmény d=12. A szimulátor segítségével állapítsa meg, hogy ezen bemeneti paraméterekre mennyi lesz a számítás eredménye. Tipp: gondolja végig, hogyan lehet a változók kezdőértékét megadni közvetlenül a szimulátor watch funkcióján keresztül vagy az előző feladatnál látott módon.
5. Javítsa ki a problémát.
6. Dokumentálja a futási eredményeket, a képletet és az elvégzett javításokat.
7. Határozza meg az algoritmus ciklusszámát és idejét a **main** címkétől indulva a **vegtelen** címkéig. Az utasítás végrehajtás órajelét állítsa 16MHz értékre.

( Dash board → Conf → Simulator → Instruction Frequency (F_{CYC}))
Időmérés: „Window → Debugging → Stopwatch”

2. Algebrai kifejezés megvalósítása

Egy adatbiztonsággal foglalkozó cég a következő algoritmus szerint számítja ki egy kifejezés értékét három bemenő paraméterből (x, y, z). Az operandusok és az eredmény is 16 bites előjel nélküli egész számok, z<5, x<1000, x>y. A kettő hatványal osztást és szorzást léptetéssel valósítsa meg.

$$F = \text{mod}_{65536} \left(((x + y) \cdot 2^z) + \left(\frac{x - y}{4} \right) \cdot x \right)$$

1. Módosítsa az előző feladat forrását úgy, hogy az az F kifejezés értékét számolja ki.
2. Definiálja az x, y, z és f változókat.
3. A program az eredményt az f változóba helyezze el.
4. Futtassa a programot különböző kezdeti értékekkel.
5. Dokumentálja a program futását a mérésvezető által megadott értékekkel.
6. Határozza meg az algoritmus megvalósításának ciklusszámát és futási idejét.

3. Algebrai kifejezés megvalósítása

Egy hőmérő szenzorból 12 bites kettes komplement érték érkezik (K). K=0 érték 20 Celsius foknak felel meg. K=1 esetén pedig 20.25 (Fix pontos tört ábrázolás, egy LSB érték 0.25 Celsius foknak felel meg).

Készítsen olyan algoritmust, amely a beérkező K értékből 16 bites előjeles egész számot csinál, úgy, hogy egy LSB pontosan 1 Celsius foknak feleljen meg és 0 Celsius fok esetén ténylegesen 0 értéket adjon. A végeredményt a W0 regiszterbe tegye.

1. Módosítsa az előző feladat forrását úgy, hogy az az új algoritmust hajtsa végre.
2. Definiálja a szükséges változókat.
3. Ellenőrizze a működést.
4. Dokumentálja a futási eredményt a mérésvezető által megadott K értékre.
5. Határozza meg az algoritmus megvalósításának ciklusszámát és futási idejét.
6. Zárja be a projektet a „File → Close All Projects” funkcióval.

A fejlesztő panel kipróbálása

Töltse le a Villamoskari oktatási portálról (edu.vik.bme.hu) az „1_labor_kijelzoproba.X.zip” projektet, majd tömörítse ki az asztalra.

A „File→Open Project...” funkcióval töltsse be ezt a projektet az MPLAB X IDE v5.50 fejlesztői környezetbe

1. Csatlakoztassa a fejlesztőpanelt a számítógép USB portjához.
2. Állítsa át konfigurációt szimulátorról hardver eszközre.

(Dashboard ablak – Properties funkció



Connected Hardware Tool: Starter Kits (PKOB)-SN:BUR....)

3. Fordítsa le és töltsse be az eszközbe a programot.

(Production – Make Program Device



)

4. A program a panelt kijelzőit teszteli.
 - A hetszegmens kijelző másodpercenként a következő számértéket jeleníti meg decimálisan (00 ... 99)
 - Potenciométer forgatásával a hetszegmens kijelző fényereje változtatható
 - Az S1 illetve S2 gomb megnyomása alatt a gomb feletti piros lett világít és az RGB LED színe megváltozik. Ha mindkét gombot egyszerre nyomva tartjuk, az RGB LED színe nem változik.

Figyelje meg a működést és próbálja ki az összes lehetőséget.

5. Zárja be a projektet a „File→Close All Projects” funkcióval.

4. LED fényerejének és színének állítása a fejlesztő panelen

Töltse le a Villamoskari oktatási portálról (edu.vik.bme.hu) az „1_labor_4_feladat.X.zip” projektet, majd tömörítse ki az asztalra.

A „File→Open Project...” funkcióval töltsse be ezt a projektet az MPLAB X IDE v5.50 fejlesztői környezetbe.

1. Fordítsa le és töltsse be az eszközbe a programot

(Production – Make Program Device



)

2. A program a potenciométer állásától függően állítja az RGB LED színét. Figyelje meg a működést. A LED erősen világít.
3. Nyissa meg a „main.s” forráskódot.
4. Módosítsa úgy a programot, hogy a fényerő alacsonyabb legyen.
5. Módosítsa úgy a programot, hogy más színek is megjelenjenek.
6. Dokumentálja a módosításokat a jegyzőkönyvben.
7. Zárja be a projektet a „File→Close All Projects” funkcióval.

A feladat során használt assembly forrás:

```
.global __reset ;kötelezően exportálandó címke, ettől kerül a reset vektor
                  ;helyére a kódunk és nem lesz semmi más a memóriában

.extern Ledinit

.text
__reset:
    ;kötelező startup kód, ha stack hivatkozás történik ezelőtt, az reset
    mov #__SP_init,w15 ; Stack pointer inicializálás
    mov #__SPLIM_init,w0 ;
    mov w0,_SPLIM ; stack limit inicializálás

main:

    mov #0, w0
    mov #0, w1
    mov #0, w2

    call Ledinit

vegtelen:

    mov 0x712,w0 ;ez a tekercs potenciális értéke (12 bites, 0..4095 tartományon)

    asr w0,#2,w0 ;összük el 4-el a bejövő értéket,
                  ;hogy a teljes tartományt használjuk

    ;R=poti értéke
    mov #1023,w1
    sub w1,w0,w1
    ;G=1023-poti

    mov w0,0x236 ;piros szín fényereje 10 biten (0: sötét, 1023: maximum)
    mov w1,0x240 ;zöld szín fényereje 10 biten (0: sötét, 1023: maximum)
    mov w2,0x24A ;kék szín fényereje 10 biten (0: sötét, 1023: maximum)

    bra vegtelen
```

5. Ellenőrző kérdések

Az alábbi ellenőrző kérdések megválaszolásához nézze át az előadáson elhangzott anyagokat, tanulmányozza a mérési útmutatót és a méréshez tartozó kiegészítő tananyagot.

- Mire használható a szimulátor "Stopwatch" funkciója?
- Mire használható a szimulátor "Watch" ablaka?
- Mire használható a szimulátor "Asynchron Stimulus" funkciója?
- Program nyomkövetése esetén mikor használjuk a "Step Into", "Step Over", és a "Run to Cursor" funkciót?
- Határozza meg, hogy mennyi lesz a d változó értéke az alábbi „C” nyelvű kifejezés végrehajtását követően, ha $a=5$. **d = a >> 2;**
- Határozza meg, hogy mennyi lesz a W2 regiszter értéke az alábbi „Asm” nyelvű kifejezés végrehajtását követően, ha $W0=5$. **asr w0, #1, w2**
- Melyik regiszter értékét módosítja az **add w0,w2,w1** utasítás?