Hardverközeli programozás oktatása a DIY Calculator segítségével

Teaching hardware programming with DIY Calculator

Varga Imre[a]

[a]Debreceni Egyetem, Informatikai Rendszerek és Hálózatok Tanszék
varga.imre@inf.unideb.hu

Absztrakt: A Debreceni Egyetemen folyó Mérnökinformatikus BSc képzés egyik alappillére a programozás oktatás, mely két részre osztható. Egyrészt magas szintű programnyelvekről tanulnak a hallgatók, másrész megismerkednek a hardverközeli programozás alapjaival. Az utóbbi témakörbe tartozó tantárgyak célja az, hogy bemutassa a hallgatóknak a számítógép részletes felépítését és működési elvét valamint megismertesse őket az assembly szintű programozás alapjaival. Közben lehetőség nyílik a tanulmányok során szerzett különböző ismeretek szintetizálására. Például megérthetik a hallgatók, hogy egy C nyelven megírt forráskódból hogyan lesz, gépi kód majd futó folyamat és eközben milyen változások játszódnak le a processzor elektronikai áramköreiben.

Ezen célok eléréséhez kiváló eszköz a didaktikai célokkal kifejlesztett úgynevezett Do It Yourself Calculator. Ez nem más, mint egy 8-bites CISC processzor alapú számológép szimulátor program assembler-rel és debugger-rel kiegészítve. Egyszerű felépítése és működési elve könnyebben megérthető, mint egy modern, komplex x86-os architektúráé, viszont segítségével megvalósítható és tanulmányozható például akár a tömbök kezelése, rekurzív alprogramhívás, megszakításkezelés, különböző perifériák meghajtása, stb. Arra szeretnék rávilágítani, hogy a hallgatók által is közkedvelt DIY Calculator milyen hasznos eszköz lehet a képzés során és milyen könnyen érhetőek el vele sikerek. Erre alapozva a valós számítógépes architektúrák felépítése és működése könnyebben megérthető.

Kulcsszavak: hardver, assembly, programozás

Abstract: One of the basic pillars of the education of Software Engineering at the University of Debrecen is the teaching of programming, which has two sections. On the one hand students study high level programming languages. On the other hand they can get to know the basics of hardware-near programming. The aim of the subjects in the latter topic is that to show how the computer build up, what are the principles of its operation, how is it working and introduce the students into the basics of assembly level programming. During studies students have possibility to synthetize their knowledge. They can get experience how a C source code became machine code and then turn into a process and during this time what kind of changes happen in the circuits of the processor.

To reach these goals an excellent tool is the didactically developed so called Do It Yourself Calculator. This is an 8-bit CISC processor based calculator simulator program integrated by assembler and debugger. Its simple structure and operation is easer understandable than the modern complex x86 architecture. Nevertheless with the help of this tool students can investigate for example how the arrays are handled, how the recursive subroutine call and interrupt handling works, and how some peripheries are driven. I would like to highlight how useful can be the popular DIY Calculator during the education. Based on the knowledge related to the virtual Calculator real architectures can be easily understandable.

Keywords: hardware, assembly, programming

1. Bevezetés

A Debreceni Egyetemen folyó Mérnök informatikus BSc képzés során a hallgatók különböző kategóriákba sorolható tantárgyakat tanulnak. Egyik ilyen kategória a mérnöki szempontból fontos, alapvető ismereteket tartalmazó tantárgyakat (pl. Fizika, Elektronika, Digitális technika, stb.) gyűjti egy csokorba. Egy másik tárgykört alkotnak a logikailag magasabb szintű informatikai rendszerekhez kapcsolódó tantárgyak, mint például a Mesterséges intelligencia, Adatbázisrendszerek, Vállalatirányítási rendszerek, stb. A kettő között egyfajta összeköttetést jelenthet a programozási tantárgykör, mely két további alcsoportra osztható. A felső szinten találhatjuk a Magasszintű programozás 1-2 illetve a Bevezetés a LabView programozásba nevű tárgyakat az alsón pedig a Hardverközeli programozás 1-2 tantárgyakat. Utóbbiaknak több szempontból is fontos szerepe van. Egyrészt az aktuális mintatanterv 2009-es bevezetése óta ez a fő színtere annak, hogy a hallgatók egyáltalán megismerkedjenek a számítógép felépítésével és működésével (ugyanis a korábbi Számítógép architektúrák tárgy kikerült a mintatantervből). Másrészt segítségükkel egységgé szintetizálható a korábban szerzett ismeretanyag, így a hallgatók átláthatják például, hogy egy C nyelven írt program futása során történő értékadás hogyan is valósul meg az egyszerű elektronikai elemekből felépülő regiszterekben lejátszódó folyamatok révén. Tehát a mérnöki képzésben kiemelkedő fontosságú hardveres ismeretek és alacsony, assembly szintű programozási képesség megszerzése a Hardverközeli programozás tantárgy hatásközében van.
Ennek oktatása során a Neumann elvű számítógépekkel történő megismerkedés válaszúthoz vezet. Vagy az idejétmúlt, de egyszerű rendszerekkel kezdjük az ismerkedést vagy egyből az aktuális technológia nagyon összetett világába ugrunk fejest. Mindkét módszernek vannak előnyei és hátrányai is. Szerintem a mai architektúrákat egy kezdő számára nagyon nehéz megérteni. Nem elég átlátni a nagyméretű általános és speciális regiszterkészletet, olyan fogalmak is hamar előtérbe kerülnek, mint a lebegőpontos egység, vektorizálás, pipelining, több szintű cache és több szintű megszakításkezelés, DMA, HyperThreading, többmagos processzorok, stb.
Ha egyszerű architektúrával akarjuk először megismertetni a hallgatókat, akkor azzal a problémával találkozhatunk, hogy az oktatásban is használható, didaktikailag megfelelő szimulátor nagyon kevés van és azok is licenszhez kötöttek. Ezen problémák orvoslására az ún. Do It Yourself Calculator kiváló gyógyír lehet.

2. A DIY Calculator

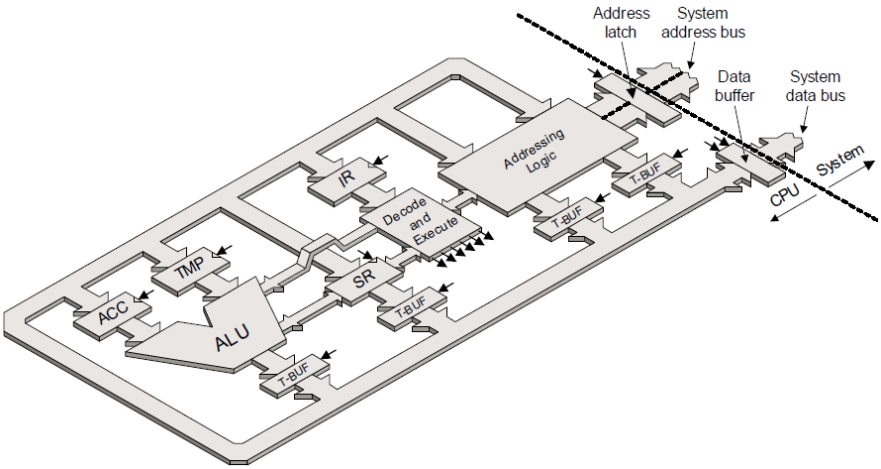
Mi is ez az eszköz? Ez nem más, mint egy egyszerű felépítésű hardver szimulátor és az ennek a programozását lehetővé tevő fejlesztői környezet. A szoftver és egyéb kiegészítők az alkotók honlapjáról [1] ingyenesen letölthető és Windows rendszeren futtatható. Fejlesztése során didaktikai szempontokat is figyelembe vettek. Felépítése és működése egyszerű, jól dokumentált. A működés során lehetősége van a programozóknak az egyes regiszterek vagy memóriarekeszek tartalmának nyomon követésére. A program az utasítások végrehajtásának a menetét, technikai részleteit részletesen, szöveges formában képes a felhasználó elé tárni. A szoftverhez létezik egy könyv [2], amely az alapoktól kezdve bevezeti a programozót abba, hogyan kell egy olyan összetettebb szoftvert megírni erre az architektúrára, amely egy egyszerű számológép funkcionalitását valósítja meg. A program dokumentációjához tartozik még egy ún. adatkönyv [3], amely az architektúra részletes elektronikai felépítésén túl a nyelv utasításkészletének részletes leírását is tartalmazza. Nézzük most meg először a szimulált hardver felépítést, majd a programozásra szolgáló saját nyelv jellemzőit.

2.1. Az architektúra

Megszokott módon a mikroszámítógépünk agya a CPU, amely a műveletek elvégéséért és döntések meghozataláért felelős. Az operatív memória tartalmazza az adatainkat és a programunkat. Az Input/Output portok lehetővé teszik, hogy a CPU kommunikáljon a külvilággal. A rendszernek imént említett komponensei a buszrendszer segítségével cserélnek információt. Nézzük hogyan is épülnek fel ezek a konkrét rendszerben!

CPU

A szimulált hardver szíve egy 8-bites CISC processzor, melynek három alrendszere az aritmetikai logikai egység (ALU), a vezérlő egység (CU) és a címző egység. Ezeket egy belső buszrendszer köti össze. Mindegyik egység működését regiszterek segítik. A CPU sematikus felépítését az 1. ábra mutatja.



1. ábra A DIY Calculator CPU-jának logikai szerkezete [3]

A regiszterek funkciójukat tekintve hasonlóak az Intel 8086-os processzor egyes regisztereihez, de elnevezésüket, méretüket és néhány apróbb jellemzőt tekintve eltérőek lehetnek attól. A virtuális hardver főbb regiszterei a következők:

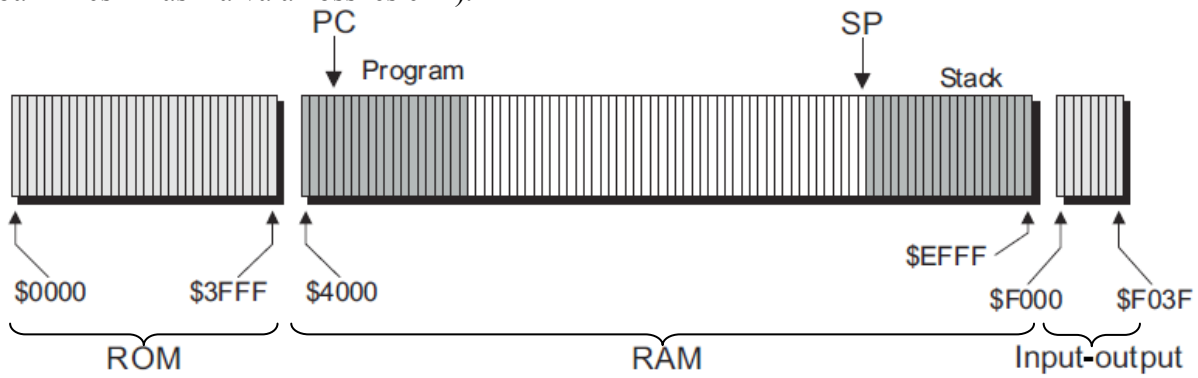
- *Programszámláló regiszter (PC):* 16 bites memóriacím tárolására szolgál, amely címen található a legközelebb végrehajtandó utasítás.
- *Utasítás regiszter (IR):* a végrehajtás alatt álló gépi utasítás 1 byte-os opkódját tartalmazza.
- *Akkumulátor (ACC):* általános munkaregiszter, mely 1 byte méretű adatot képes tárolni és a legtöbb művelet egyik operandusa és eredménye is ide kerül.
- *Státusz regiszter (SR):* a végrehajtást befolyásoló 5 jelzőbitet tartalmazó (logikailag 1 byte-osnak tekintett) regiszter. Bitjeinek jelentése a következő. I (interrupt): be van-e állítva megszakítás figyelés, O (overflow): a legutóbbi művelet során történt-e túlcsordulás, N (negative): az ACC tartalma előjeles egész számábrázolást tekintve negatív-e, Z (zero): az ACC tartalma vagy a legutóbb végrehajtott művelet eredménye nulla-e illetve összehasonlító művelet egyenlőséget adott-e, C (carry): a legutóbb végrehajtott művelet során keletkezett-e átvitelbit illetve összehasonlító művelet eredménye az e hogy az ACC tartalma nagyobb, mint az összehasonlítás operandusa.
- *Index regiszter (X):* indexelt címzés esetén a címeltolás 2 byte-os értékét tároló regiszter.
- *Verem mutató (SP):* a memória azon címét tartalmazó 2 byte-os regiszter ahová a rendszer verembe betenni kívánt elem kerül illetve veremből történő kivétel esetén az SP által tárolt címnél egyel nagyobb memóriacímen lévő érték (verem teteje) kerül kivételre.
- *Megszakítási vektor (IV):* a megszakítási rutin RAM-beli címét tároló 16 bites regiszter.
- *Átmenti programszámláló A és B regiszter (TPCA, TPCB):* abszolút, indexelt és indirekt címzéshez szükséges 2 byte-os ideiglenes tároló.

- *Átmeneti regiszter (TMP):* kétoperandusú művelet esetén a második operandust átmenetileg tartalmazó 8 bites regiszter.
- *Address latch (MAR):* az itt tárolt cím kerül a címbuszra, azaz olvasás esetén erről a memóriacímről olvasunk, írás esetén erre a címre kerül a tárolandó adat.
- *Data buffer (MDR):* olvasás esetén az adatbusz tartalma ide kerül, írás esetén az MDR tartalma kerül az adatbuszra.

A hivatalos adatkönyv [3] CPU egyes részeinek felépítését pontosan leírja, néha egészen logikai kapu szintig lemenve. Ha nem csak a szerkezeti felépítés érdekel bennünket, akkor a logikai működésen túl az egyes utasítások végrehajtása során különböző síneken mérhető feszültségek idődiagramjait is megtekinthetjük órajeltől órajelre. Ez hihetetlenül fontos lehet mérnöki szempontból és így a képzés nélkülözhetetlen részét alapozhatjuk erre az eszközre.

Memória

A hardver mások fontos eleme a memória, mely a Neumann elveknek eleget téve azt adatainak és a programunk tárolására szolgál. A DIY Calculator csak olvasható (ROM) és írható-olvasható (RAM) memóriát is tartalmaz. A memóriarekeszek byte-onként címezhetőek, 16 bites címekkel. Így egy 64kB méretű címterünk van, aminek az elején (16kB) ROM található, majd a (44kB méretű) RAM helyezkedik el (2. ábra). Az utolsó 4kB méretű tartományban ’memory mapped I/O’ technikát alkalmazva input/output címeket találhatunk (bár nincs kihasználva az összes cím).



2. ábra: A 64kB-os címtér felosztása

I/O protok

Ahogy már említettük ’memory mapped I/O’ révén ugyanaz a címbusz használható memóriarekesz és I/O port hivatkozásra. 32 input ($F000-$F01F) és 32 output ($F020-$F03F) port érhető el. Ezekre a hamarosan tárgyalandó perifériák kapcsolódnak.

Buszrendszer

Itt is, mint általában 3 busz található: cím-, adat- és vezérlő busz. A kétirányú adatbusz 8 bit széles illeszkedve az ACC méretéhez. A 64kB-os címtérnek megfelelően a címbusz 16 sínt tartalmaz. A vezérlő busz mindössze néhány vezetékből áll. Ilyen az írási (*write*) illetve olvasási (*read*) engedélyező vonal, a hagyományos *reset* vonal, a szinkronizálásra szolgáló órajelet biztosító *clock* vonal továbbá a megszakítási kérelem (*IRQ*) és megszakítási nyugta (*IACK*) vezeték.

Perifériák

A CPU az I/O portokon és a buszrendszeren keresztül különböző perifériákat ér el. Az alapértelmezett periféria az ún. Front panel, mely egy LCD kijelzőt, egy 6 elemű LED sort, egy (átcímkézhető) számológép billentyűzetet (keypad) valamint néhány vezérlő gombot (On/Off, Reset, Run, Step) tartalmaz. Ezen kívül használhatunk egy szöveges terminált, egy teljes qwerty billentyűzetet, és az ún. Workbench nevű eszközt. Utóbbin helyet kapott 2 darab 8 bites kapcsolósor, egy 8 bites LED sor, egy-egy dekódolt szimpla és dupla valamint egy dekódolatlan 7 szegmenses kijelző. A Calculator segítségével tehát a hallgatók az ilyen egyszerűbb hardver elemek meghajtását is elsajátíthatják.

2.2. A DIY Calculator assembly nyelve

Az előbbiekben áttekintett hardver programozására assembly szinten van lehetőség. A program ASCII karakterekből épül fel. A nyelv betűnek tekinti az angol abc karaktereit és az aláhúzás (_) jelet, a kis és a nagy betűket nem különbözteti meg. A numerikus konstansok megadhatóak decimális (pl.: 90), bináris (pl.: $5A) vagy hexadecimális (pl.: %1011010) alakban. Nevesített konstans fordítási direktívával hozható létre. Egy forráskód maximum egy darab '1 címes' utasítást tartalmazó sorokból áll, ahol egy utasítás 4 elemből épül fel: címke, mnemonik, operandus, megjegyzés.

- *Címke:* az utasításokra történő hivatkozáshoz szükséges. Betűvel kezdődik és betű vagy számjegy karakterrel folytatódó maximum 8 karakter hosszúságú azonosító kettősponttal (:) lezárva. Mindig egy RAM címet jelent, azt helyettesíti. Nem kötelező elem.
- *Mnemonik:* az utasítás/operátor néhány kerekteres kulcsszava (pl. LDA: LoaD Accumulator).
- *Operandus:* a kétoperandusú operátor második operandusa (az első mindig az ACC-ben van). Lehet adat vagy cím is. Nem minden utasításban szükséges.
- *Megjegyzés:* kettős kereszt (#) karaktertől a sor végéig található programrészt az assembler figyelmen kívül hagyja.

A nyelv 47 végrehajtható és 7 fordítónak szóló utasítást (direktíva) ismer. Az előbbiek besorolhatóak néhány kategóriába: mozgató utasítás, bitenkénti műveletek, aritmetikai műveletek, vezérlésátadó és egyéb utasítások. Utóbbiak mnemonikja pont (.) karakterrel kezdődik. Memóriaterület foglalás (például statikus változó részére) direktívával történik. A betöltés és a címfordítás kezdőcímének a megadása szintén direktíván keresztül valósul meg.

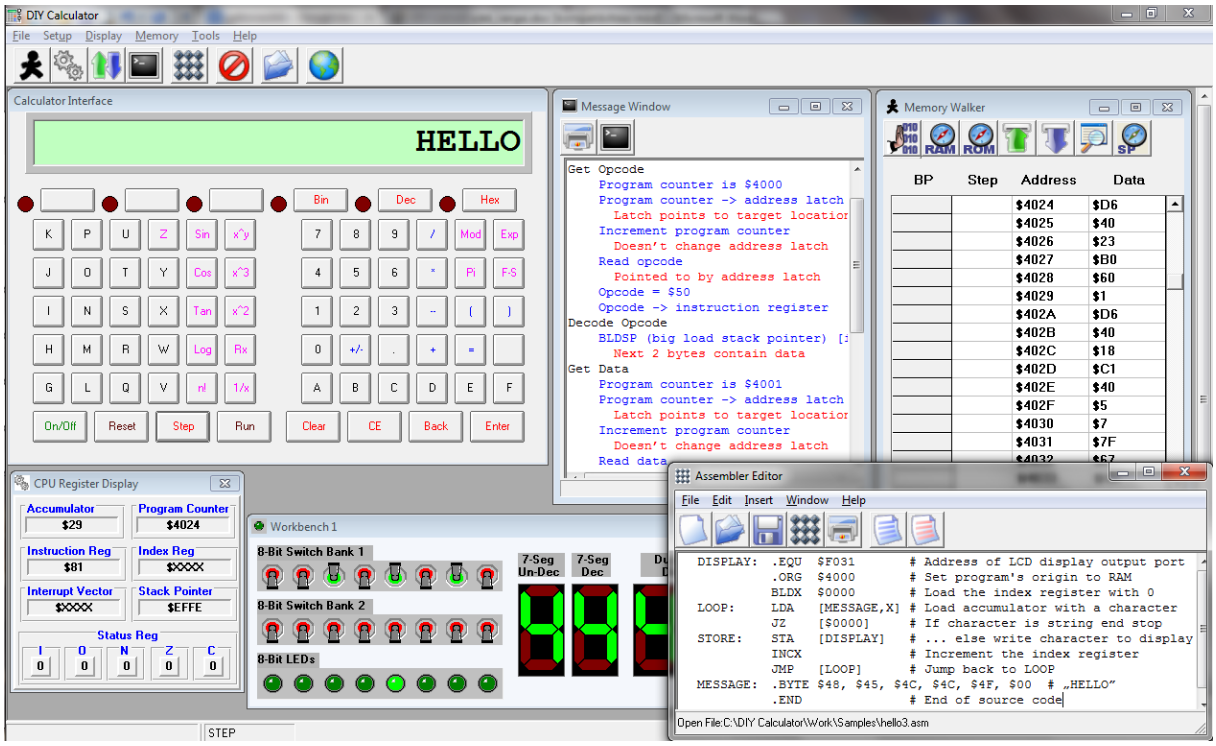
A CPU 7 különféle címzési módot tud megvalósítani, hogy egy adatra hivatkozzon:

- *rejtett (implied):* a műveletnek nincs szüksége operandusra (vagy csak egyre, de az az ACC-ben van), így ez nem is 'valódi' címzés. (pl.: INCA)
- *alap/nagy kódba épített (standard/big immediate):* memóriában az operandus kód után közvetlenül találjuk az 1 vagy 2 byte méretű operandust. (pl.: ADD $10, BLDSP $EFFF)
- *alap/nagy közvetlen (standard/big absolute):* RAM-ban az utasítás opkódját egy 2 byte-os memóriacím követi, amely megmondja hol található a memóriában az 1 vagy 2 byte méretű operandus. (pl.: OR [$4100])
- *indexelt (indexed):* a műveleti kód után egy címet találunk, ehhez hozzáadva az index regiszter (X) tartalmát megkapjuk azt a címet, ahonnan az operandust vennünk kell. (pl.: SUB [$4100,X]
- *közvetett (indirect):* az opkód utáni két byte-on egy cím található, amely megmutatja, hol van az a másik cím, ami meghatározza az operandus értékét. (pl.: JMP [[$4100]])
- *előzetesen indexelt közvetett (pre-indexed indirect):* a műveleti kód utáni 2 byte-os címhez hozzáadva az index regiszter értékét megkapjuk azt a címet, ahol megtalálható az operandus címe. (pl.: LDA [[$4100,X]])
- *utólagosan indexelt közvetett (indirect post-indexed):* az utasítás gépi kódját a memóriában egy cím követi, mely egy másik 2 byte-os cím helyét határozza meg. Az ezen a címen található értékhez hozzáadva az index regiszter értékét megkapjuk azt a címet, ahol megtalálható az operandus értéke. (pl.: STA [[$4100],X])

Amint látható ez az egyszerű kis eszköz is egészen összetett viselkedést tud megvalósítani. Bár a neve alapján csak egy egyszerű számológép hardvernek gondolhatnánk, azért képességei lehetőséget nyújtanak ahhoz, hogy akár bonyolultabb programozási eszközöket és adatszerkezeteket is megvalósítsunk vele. Készült már például BASIC interpreter is ezen a nyelven.

2.3. A fejlesztői környezet

A program forráskódja a fejlesztői környezet editorában (is) megírható. `.asm` kiterjesztésű file-ként jelenik meg a PC-n. Ebből képez az assembler (szöveges formátumú) gépi kódot `.ram` kiterjesztéssel. Ez már közvetlenül betölthető a DIY Calculator RAM-jába, majd a 'Run' gomb megnyomásával futtatható is a program. Lehetőség van továbbá a futtatásra utasításonként is. Hogy jobban megérthető és következő legyen a fordítás menete az assembler a fordítás részleteiről (forrás és gépi kód összerendelés, címkéknek megfelelő címek listája, stb.) további log file-okat is készít `.lst` és `.rad` kiterjesztéssel.
Bár az assembler szintaktikai hiba esetén nem túl bőbeszédű, a fejlesztői és szimulációs környezet nyomkövető eszközök egész garmadáját biztosítja a programfejlesztő hallgatók rendelkezésére (lásd 3. ábra). Az úgynevezett Memory Walker-rel byte-ról byte-ra megtekinthetjük a teljes memória tartalmát, töréspontokat helyezhetünk el benne. A 'CPU regiszter display' valós időben mutatja a fontosabb regiszterek (ACC, IR, PC, SR, X, IV) aktuális tartalmát a futás során. Az 'I/O port display' megmutatja néhány ki és bemeneti puffer pillanatnyi tartalmát. A 'Message window' nevű eszköz a futás közben történő eseményeket írja le szövegesen szinte órajel szintre lebontva az időt. Ha a futás túl gyors a virtuális hardver órajelét is tudjuk állítani szoftveresen. Mindezek az eszközök nagyon hasznosak ahhoz, hogy a hallgatók megértsék a háttérben folyó mérnöki szemmel fontos eseményeket. Segítségükkel könnyebben rájöhetnek, hogyan kell optimálisabb a programot írni akár magas nyelven is. Mindezekre valós hardver esetén egyáltalán nincs lehetőség. Vegyük még ehhez hozzá, hogy a DIY Calculator felépítéséhez és működéséhez részletes dokumentáció érhető el [3], ami nem minden mai valós architektúra esetén mondható el. A dokumentáció annyira részletes, hogy két csapat is megvalósította a hardvert FPGA segítségével.
A 3. ábra a számológép fő egységét (a front panelt), a debug valamint I/O eszközök egy részét mutatja be. Az ábra jobb alsó sarkában pedig az assembler editora található. Ebben egy példát láthatunk a DIY Calculator programra, mely a „HELLO" sztringet írja ki a front panel LCD kijelzőjére.



3. ábra: A DIY Calculator fejlesztői- és futtatókörnyezetének egy része

3. Összegzés

A mérnökinformatikus képzés fontos feladata, hogy megismertessük a hallgatókkal, hogyan is épül fel a számítógép mérnöki szemmel illetve, hogy a programozással kapcsolatos és az elektronikai ismeretek közötti rést áthidaljuk. Ennek a célnak az eléréséhez nagyszerű eszköz lehet a DIY Calculator nevű virtuális hardver. Felépítése egyszerű, a dokumentációk alapján könnyen megérthető. Programozása saját assembly-szintű nyelven történik. A használat közben könnyebben megérthetik a hallgatók hogyan is működik hardverközeli szinten az, amit például egy C program esetén akár nap, mint nap használnak. Gondoljunk itt a következőkre: mutatók, tömbök elemeire hivatkozás, sztringek tárolás, eljáráshívás, érték visszaadás függvény esetén, paraméterátadás, lokális változók dinamikus élettartam kezelése, rekurzió, veremkezelés, megszakításkezelés, stb. Emellett – mellékhatásként – néhány külső elektronikai eszköz (pl.: kódolt/kódolatlan 7 szegmenses kijelző, LED sor, kapcsolósor) meghajtását is elsajátíthatják. A hallgatók ¾ része vallja azt, hogy a Calculator segített neki a magas szintű programozási ismereteinek az elmélyítésében is.
Természetesen a DIY Calculator nem helyettesíti a valós processzorok (x86, ARM, stb.) bemutatását és ezek programozásának az elsajátítását. Azonban kezdők számára első lépésnek kiváló, mivel a processzorműködés, programfuttatás valós időben monitorozható a fejlesztői környezet és a hardver szimuláció nyomkövető eszközrendszerével, könnyebbé téve a megértést és az elsajátítást. Néhány éves tapasztalat alapján azt mondhatom, hogy használata az képzés hatásfokát javítja.

Irodalomjegyzék

[1] Online: http://www.clivemaxfield.com/diycalculator (legutóbbi megtekintés: 2014. június 14.)

[2] Clive „Max” Maxfiled és Alvin Brown: The definitive guide to How Computers Do Math, Wiley (2005).

[3] Clive „Max” Maxfiled és Alvin Brown: The official DIY Calculator Data Book, (2005)

[4] Clive „Max” Maxfiled és Alvin Brown: Introducing the Virtual DIY Calculator, *Everyday Practical Electronics* **34**, No. 10 (2005), 694-696.