

**Szegedi Tudományegyetem**

**Informatikai Intézet**

**Az emulátor fejlesztési folyamat bemutatása a  
Nintendo Game Boy konzolon**

Diplomamunka

*Készítette:*

**Krizsák Tibor**

informatika szakos

hallgató

*Témavezető:*

**Dr. Tanács Attila**

egyetemi adjunktus

Szeged

2018

# Tartalomjegyzék

Feladatkiírás . . . . .	3
Tartalmi összefoglaló . . . . .	4
Bevezetés . . . . .	5
<b>1. Az emulátorokról</b>	<b>6</b>
1.1. Az emulátor fogalma . . . . .	6
1.2. Az emulátorok típusai . . . . .	6
Nyilatkozat . . . . .	8
Köszönetnyilvánítás . . . . .	9
Irodalomjegyzék . . . . .	10

# Feladatkiírás

A Nintendo Game Boy egy 1989-ban bemutatott, 8 bites kézi videojáték-konzol. A konzolban egy Zilog Z80 (az Intel 8080 "utódja") processzor működik, kiegészítve néhány specifikus utasítással.

A feladat egy emulátor fejlesztési fázisainak bemutatása a Nintendo Game Boy hardveren, valamint a kész emulátor tesztelése. Az emulátor Rust nyelven legyen megvalósítva, illetve a tesztelés a más Zilog Z80 emulátor fejlesztők által készített teszt ROM-okon történjen.

# Tartalmi összefoglaló

- **A téma megnevezése :**

Egy emulátor fejlesztési fázisainak bemutatása a Nintendo Game Boy hardveren keresztül, Rust nyelven implementálva.

- **A megadott feladat megfogalmazása :**

A feladat egy Nintendo Game Boy emulátor implementálása, és fejlesztési fázisainak bemutatása. A bemutatás során a CPU utasításait, a GPU renderelésének működését, a memóriakezelést, és a megszakítás-vezérlést kell érinteni, illetve az egyéb kisebb, de a működéshez elengedhetetlen megoldások is megemlítésre kerülnek. Ahhoz, hogy ezek megfelelően működjenek, a CPU frekvenciájára, illetve a képfrissítési gyorsaságra is tekintettel kell lenni.

- **A megoldási mód :**

Az emulátor fejlesztő közösség által összegyűjtött - reverse-engineered - információkra, illetve a processzor gyártója által kiadott technikai dokumentációra hagyatkozva felépítettem és implementáltam a CPU struktúráját, uasításkészletét, majd a többi modult, részegységet. Meghatároztam a modulok közti kommunikációt, időzítéseket, adatfolyamot. A videojáték-, illetve teszt ROM-ok byte-jait sorra beolvasva az emulátor meghatározza a megfelelő műveletet, meghatározott időközönként renderel, illetve kezeli a megszakításokat.

- **Alkalmazott eszközök, módszerek :**

Az emulátor Linux rendszeren, Rust nyelven került implementálásra, a `rustc` fordító, illetve a `cargo` package manager segítségével. A rendereléshez a `minifb` libraryt használtam, ami egy nagyon egyszerű framebuffer használatát teszi lehetővé. A fejlesztésre került egy debugger eszköz, illetve egy memória térkép eszköz is, ami nagyban megkönnyítette a hibakeresést.

- **Elért eredmények :**

Az implementált emulátor képes futtatni Memory Banking nélküli videojáték ROM-okat, az inputra az elvárásoknak megfelelően reagálva. A processzor műveletek és a renderelés az eredeti konzollal megegyező eredményt adnak. A közösségi Game Boy teszt ROM-ok szinte mindegyikét sikerrel végrehajtja.

- **Kulcsszavak :**

Nintendo, Game Boy, emulátor, fejlesztés, Rust

# Bevezetés

A számítástechnikában az emuláció fogalma nem új keletű. Különböző területeken, különféle problémák megoldására használnak emulátorokat, ugyancsak különböző okokból. A nyomtatóktól kezdve, a DOS-kártyákon keresztül, a többmagos rendszertervezésen át egészen a videojáték konzolokig terjed a paletta - nem túlzás azt állítani, hogy az emulátorok ott vannak a mindennapjainkban.

Ezen diplomamunka a videojáték konzolok emulátorainak fejlesztésére fókuszál. Többféle cél állhat a háttérben ha valaki ilyen emulátor fejlesztésére adja a fejét: a régi hőskorbéli konzolok digitális megőrzése vagy életre keltése, későbbi szoftverfejlesztés az emulált hardveren, esetleg hobbiként. Az utóbbi évek tendenciája azt mutatja hogy ez utóbbi ok egyre gyakoribb - az emulátor fejlesztői közösség napról napra nagyobb és aktívabb, szokások és kisebb fejlesztői folklór alakult ki az emulátor készítését illetően - a dolgozat ennek bemutatására helyezi a hangsúlyt.

Az emulátor fejlesztés szemléltetése Nintendo Game Boy kézi videojáték konzolon keresztül fog történni, amely a maga idejében egy igazán sikeres konzol volt, és tulajdonképpen kultusz épült köré. A 8 bites architektúrájából adódóan kevésbé bonyolult felépítéssel rendelkezik, népszerűségéből adódóan jól dokumentált, így az emulálásának implementációjához nincs szükség túl sok reverse-engineering gyakorlatra.

A dolgozat első néhány fejezetében az emulátorokról, a Nintendo Game Boy hardveréről, specifikációjáról, illetve a későbbi fejlesztés workflow-járól fog szó esni. Ezekben a fejezetekben van megfogalmazva, illetve leírva az, hogy pontosan mi az az emulátor, milyen hardver emulációjáról van szó, és hogy az emuláció teljes implementálásáig milyen pontokon keresztül vezet az út. A következő nagyobb logikai egység az implementáció. Ennek részeként először bemutatásra kerülnek az alkalmazott eszközök, technológiák, majd az emulátor pontos és elvárt specifikációjának leírását az igazi implementációs szakasz követi.

A processzor modellezése a regiszterek, flagek, és egyéb jellemzők megtervezésével kezdődik, majd következő lépésként az utasításkészlet megvalósításával folytatódik. A CPU-hoz szorosan kapcsolódó memória ez után kerül tárgyalásra. A memória ismertetése után az időzítők, majd a PPU felépítése és működése szerepel. Az implementáció ezen pontján a boot ROM már futtathatóvá válik, erről is esik majd néhány szó. A fejlesztési részt a joypad jellemzői és megoldásai zárják.

A dolgozat zárásaként bemutatásra kerül az emulátor használata, illetve a fejlesztésből adódó dependenciák, majd végül a teszt ROM-ok jellemzői, futtatásuk, és a futtatási eredményeik.

# 1. fejezet

## Az emulátorokról

– Definíció, működés, fajtái, jelene, jövője, jelentősége

Az utóbbi évtizedekben végbement – és jelenleg is tartó – technikai fejlődés következményeként rendkívül gyors a technológiai elavulás. Ennek következményeképp az eszközök életciklusa megrövidül, értékük rohamosan csökken. Gyakran előfordul azonban, hogy szükség van a régi *legacy* rendszerekre, vagy elengedhetetlen a visszafelé kompatibilitás, esetleg szeretnénk az adott hardvert a számítástechnikai jelentősége miatt valamilyen formában megőrizni, használhatóvá tenni. Az emulátorok ezekre a problémákra igyekeznek megoldást kínálni – persze rendkívül sok egyéb felhasználási terület mellett.

### 1.1. Az emulátor fogalma

Definíció szerint olyan hardvert vagy szoftvert nevezünk **emulátornak**, amely lehetővé teszi, hogy egy számítástechnikai rendszer (szokás ezt *host*-nak nevezni) úgy viselkedjen, mint egy másik számítástechnikai rendszer (ez pedig a *guest*). Jellemzően az emulátor a *host* rendszer számára teszi lehetővé olyan szoftver futtatását vagy periféria használatát, amely a *guest* rendszerhez lett kifejlesztve. Röviden megfogalmazva az emulátor egy olyan hardver vagy szoftver, ami egy másik eszközt vagy programot emulál, imitál.

### 1.2. Az emulátorok típusai

Az emulátorok többsége csak a hardver architektúrát emulálja – ha operációs rendszer vagy egyéb szoftver is szükséges az emuláláshoz, akkor azt is biztosítani kell. Ebben az esetben az operációs rendszert és a szoftvert *interpretálni* (értelmezni) fogja az emulátor. A gépi kód *interpreteren* kívül azonban az emulátornak tartalmaznia kell a *guest* hardver minden lehetséges jellemzőjét, és viselkedését is virtuálisan: ha például egy adott memórhelyre való írás befolyásolja azt, hogy mi jelenik meg a képernyőn, úgy azt is emulálni kell. Habár lehetne az emulációt extrém részletességgel, atomi szinten végezni – például az áramkör adott részei által kibocsátott pontos feszültség-ingadozás emulálásával, stb. – ez egyáltalán nem gyakori, az emulátorok általában megállnak a dokumentált hardver specifikáció, és digitális logika szimulációjának szintjén.

Némely hardver hatékony emulálásához extrém pontosság szükséges: az óraciklusokat,

nem dokumentált jellemzőket, kiszámíthatatlan analóg elemeket, és *bugokat* mind-mind implementálni kell. A klasszikus otthoni számítógépek esetében (például a Commodore 64) ez hatványozottan igaz, mert az ezekre a hardverekre írt szoftverek gyakran kihasználtak alacsonyszintű programozási trükköket, melyeket főként a videojáték programozók és a *demoscene* fedeztek fel.

In contrast, some other platforms have had very little use of direct hardware addressing, such as an emulator for the PlayStation Vita. In these cases, a simple compatibility layer may suffice. This translates system calls for the emulated system into system calls for the host system e.g., the Linux compatibility layer used on \*BSD to run closed source Linux native software on FreeBSD, NetBSD and OpenBSD. For example, while the Nintendo 64 graphic processor was fully programmable, most games used one of a few pre-made programs, which were mostly self-contained and communicated with the game via FIFO; therefore, many emulators do not emulate the graphic processor at all, but simply interpret the commands received from the CPU as the original program would. Developers of software for embedded systems or video game consoles often design their software on especially accurate emulators called simulators before trying it on the real hardware.

# Nyilatkozat

Alulírott ..... szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet ..... Tanszékén készítettem, ..... diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat / diplomamunkámat a Szegedi Tudományegyetem Informatikai Intézet könyvtárában, a helyben olvasható könyvek között helyezik el.

Szeged, 2018. február 27.

.....  
aláírás



# Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani **X. Y-nak** ezért és ezért . . .

# Irodalomjegyzék

- [1] J. L. Gischer, The equational theory of pomsets. *Theoret. Comput. Sci.*, **61**(1988), 199–224.
- [2] J.-E. Pin, *Varieties of Formal Languages*, Plenum Publishing Corp., New York, 1986.