Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Generování linuxového jádra pro embedded zařízení

Bc. Vojtěch Hrdina

Diplomová práce

2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2018/2019

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Vojtěch Hrdina**
Osobní číslo:	**I16218**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Generování linuxového jádra pro embedded zařízení**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Cílem diplomové práce je generování linuxového jádra pro embedded zařízení. Jako testovací platformu použijte Raspberry Pi 3, Beaglebone Black a Orange Pi ZERO. Popište a testujte frameworky OpenADK a Buildroot, popř. další vhodné. Součástí práce budou benchmarky včetně původních distribucí (Raspbian, Debian). V teoretické části bude popsán způsob konfigurace a kompilace linuxového jádra včetně postupu instalace jádra na tyto platformy.

Rozsah pracovní zprávy: **60**
Rozsah grafických prací: **10**
Forma zpracování diplomové práce: **tištěná**

Seznam doporučené literatury:

BOVET, Daniel P. a Marco. CESATI. Understanding the Linux kernel. 3rd ed. Sebastopol, CA: O'Reilly, c2006. ISBN 978-0596005658.
CORBET, Jonathan., Alessandro. RUBINI a Greg. KROAH-HARTMAN. Linux device drivers. 3rd ed. Sebastopol, CA: O'Reilly, 2005. ISBN 0-596-00590-3.
OpenADK – Open Source Appliance Development Kit [online]. 2017 [cit. 2017-10-04]. Dostupné z: https://openadk.org/.

Vedoucí diplomové práce: **Ing. Miroslav Dvořák, Dipl.tech.**
Katedra informačních technologií

Datum zadání diplomové práce: **30. října 2018**
Termín odevzdání diplomové práce: **18. května 2019**

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 18. 5. 2019

Bc. Vojtěch Hrdina

PODĚKOVÁNÍ

Tímto bych rád poděkovat panu Ing. Miroslavovi Dvořákovi, Dipl.tech. za jeho trpělivost a čas, který mi věnoval. Za jeho pomoc, připomínky a rady. Dále bych rád poděkoval své rodině a přátelům, kteří mě celou dobu mého studia podporovali a doprovázeli.

ANOTACE

Cílem diplomové práce je generování linuxového systému pro embedded zařízení za použití frameworku Buildroot a OpenADK. Jako testovací platformy byly zvoleny Raspberry Pi 3, BeagleBone Black a Orange Pi ZERO. Výsledné systémy budou porovnány s oficiálně doporučenými systémy daných zařízení.

KLÍČOVÁ SLOVA

embedded, vestavěný systém, linux, kernel, jádro, Buildroot, OpenADK, Raspberry Pi, Orange Pi, BeagleBone, libc, framework

TITLE

Linux kernel generation for embedded device

ANNOTATION

The aim of this work is to generate a Linux system for embedded devices using the Buildroot and OpenADK frameworks. Raspberry Pi 3, BeagleBone Black and Orange Pi ZERO were selected as test platforms. The resulting systems will be compared with the officially recommended equipment systems.

KEYWORDS

embedded, linux, kernel, Buildroot, OpenADK, core, OepnADK, Raspberry Pi, Orange Pi, BeagleBone, libc, framework

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK A ZNAČEK

ARM – Advanced RISC Machine

RISC – Reduced Instruction Set Computer

CPU – Central Processing Unit

GPU – Graphics Processing Unit

SoC – Systém on Chip

API – Aplication Programming Interface

BLE – Bluetooth Low Energy

PC – Personal Computer

USB – Universal Serial Bus

I2S – Inter-IC Sound

IR – Infared light

PWM – Pulse Width Modulation

SPI – Seriál Peripheral Interface

I2C – Two Wire Interface

UART – Universal Asynchronous Receiver-Transmitter

CSI – Construction Specifications Institute

MMC – Multi Media Card

eMMC – embedede Multi Media Card

PMIC – Power Management Integrated Circuit

LDO – Low-drop regulator

POE – Power Over Ethernet

RDP – Remote Desktop Protocol

TERMINOLOGIE

open source – program s dostupným zdrojovým kódem

GNU C Library – standardní knihovna jazyka C

ISO C – standard knihovny jazyka C

Makefile – soubor určující způsob překladu a definuje závislosti mezi zdrojovými soubory

root – uživatel s administrátorskými oprávněními k systému,

filesystem – souborový systém

kernel – jádro operačního systému Linux

Open Accessory Kit – sada nástrojů

cross-compiler – kompilátor vytvářející spustitelný soubor pro odlišnou architekturu

XRDP – open-source implementace Microsoft RDP

systemctl – řízení služeb systému

benchmark – programy k posouzení relativního výkonu

shell – textové uživatelské rozhraní

bash – „*Bourne again shell*“ – typ shellu

deamon – počítačový program běžící na pozadí

watchdog - počítačová periferie, která resetuje systém při jeho zacyklení, nebo zaseknutí

ÚVOD

Diplomová práce se bude zabývat generováním Linuxových operačních systémů pro malá přenosná zařízení za pomocí frameworků Buildroot a OpenADK. Nedílnou součástí diplomové práce bude rozsáhlejší uvedení do koncepce operačního systému Linux a porovnání běžné desktopové verze s verzí pro embedded zařízení.

V úvodní části práce bude popsána architektura ARM pro bližší obeznámení s testovací platformou Raspberry Pi3, BeagleBone Black a Orange Pi ZERO. Jednotlivé platformy budou také podrobně popsány.

Před zahájením procesu tvorby generování linuxového systému budou představeny frameworky, u nichž bude popsán základní postup práce a tipy na lepší efektivitu jejich použití.

Konfigurace bude začínat na nejnižší hardwarové úrovni od řazení dat v paměti přes konfiguraci jádra systému a implementaci ovladačů pro dané zařízení. V pokročilejší fázi budou konfigurovány systémové a uživatelské programy pro samotnou práci se zařízením.

V poslední části diplomové práce bude provedeno porovnání vygenerovaných systémů s dodávanými alternativami od výrobců daných zařízení.

Cílem práce je pomocí frameworků vytvořit efektivnější systémy než dodávané od výrobců daných zařízení.

1 Embedded zařízení

Zařízení určena k jedné, nebo několika málo funkcím jsou označována jako embedded systémy. S nejčastějším využitím se s nimi setkáme v průmyslových zařízeních, automatizaci, spotřební elektronice, ale i v chytrých domácnostech a automobilech. Základní jednotkou bývá mikrokontroler, což je procesor s operační pamětí a periferními prvky. Typické periferní prvky jsou kromě sériových linek, sloužících k propojení zařízení, také ethernet řadiče pro síťovou komunikaci, PWM generátory pro pulzně modulované signály, analogové a digitální převodníky, USB řadiče, šifrovací jednotky, čítače a časovače pro měření nebo generování časových událostí, ale i detektory poklesu napájení a obvody typu watchdog pro kontrolu správného běhu celého systému.

Embedded zařízení mají tendenci přiblížit se plnohodnotným počítačovým systémům. Vícejádrové procesory s paralelním zpracováním instrukcí, relativně velké množství operační paměti a nespočet periferií zajišťují širokou škálu možností užití. Dnes stále nezanedbatelnými rozdíly oproti běžným systémům, typu osobní počítač, jsou řádově méně výkonné procesory určené jen na omezený typ úloh, rozdělená paměť pro kód a data a cílení především jen na nízkoúrovňové periferie, jako jsou sériové sběrnice, radia, rozhraní pro minimalistické zobrazovače a další.

Nejčastěji zastoupený programovací jazyk pro mikroprocesory embedded systémů je C/C++ a assembler. Pro nízkoúrovňové programování se vývojář neobejde bez znalosti přerušovacího podsystému, mapování paměti a podrobné dokumentace samotného zařízení.

Stejně jako zařízení určené pro jednu, nebo jen několik funkcí, tak i operační systémy pro embedded zařízení by měly být optimalizovaná pro konkrétní zařízení a daný úkol. [15]

1.1 Raspberry Pi 3

Mezi nejznámější embedded zařízení patří Raspberry Pi vyvinuté britskou nadací Raspberry Pi Fundation. V diplomové práci byla pro účely testování použita verze tři, která byla uvedena do prodeje v únoru 2016. Její cena byla stanovena na 35$ (800Kč).

Oproti svému předchůdci byla deska osazena 64bitovým[1] CPU RISC ARM Cortex-A53, který je pro svůj výkon používán i v multimediálních zařízení pro běžné úlohy, jako je prohlížení webových stránek, přehrávání videí a provádění výpočetně méně náročných úloh.

Integrované grafické jádro střední třídy Mali-T720 s taktem 550 – 650MHz je jako první založeno na základě architektury Midgard, doposud používané pouze u high-end čipů. Grafické jádro je optimalizováno pro zařízení s operačním systémem Android a podporuje OpenGL ES 3.0 a RenderScript. Ani přes využití plného potenciálu se nehodí na hraní her a graficky náročné aplikace.

Pro procesor a grafické jádro je dostupná operační paměť o velikosti 1 GB. Je zde integrováno ethernetové rozhraní s maximální teoretickou propustností až 300 Mbit/s.

Největší přínos oproti předešlému modelu je integrovaný dvoupásmový Wi-Fi modul 2.4/5GHz s podporou standardu 802.11ac a Bluetooth modul 4.2 BLE.

Na desce jsou dostupné 4 USB porty ve verzi 2.0. Další dostupné rozhraní je MIPI pro video vstup, se specifikací CSI, vhodný pro kamerový senzor. Video výstup je zde kromě digitálního HDMI i kompozitní – analogový. Video je také dostupné přes rozhraní MIPI, se specifikací DSI[2], které je určené pro miniaturní LCD. Analogové audio je zde dostupné přes 3,5mm jack a digitální přes HDMI. Pozdější verze desky disponuje audio výstupem i přes I2S sběrnici.

Zařízení nedisponuje vnitřním uložištěm, ale pouze microSD slotem, který splňuje standard SDXC s podporou maximální velikosti 2TB na souborovém systému exFAT. Mimo bootování z předurčeného slotu je možné bootování přes rozhraní USB, nebo ze sítě.

[1] Architektura PC založena na 64bitových registrech a sběrnicích (adresové, datové).
[2] Display Serial Interface

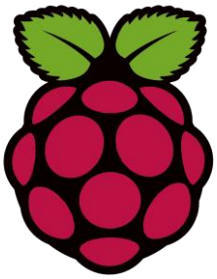
V neposlední řade je jeden z klíčových prvků periferní nízkoúrovňové zařízení GPIO (40 pinů), které je softwarově nastavitelné pro vstup, nebo výstup k širokému rozsahu použití. Alternativní integrované funkce jsou PWM, SPI, I2C a UART. [4]

Zařízení je napájeno napětím 5V pomocí microUSB konektoru. Alternativní možnost je napájet zařízení přes GPIO piny, nebo přes Ethernet (PoE). Rozměry zařízení jsou 86x57x17mm. [4]

Obrázek 1: Raspberry Pi3 [4]

1.1.1 Raspbian

Systém vychází z distribuce Debian a je optimalizovaný pro zařízení Raspberry Pi. Obsahuje sadu základních programů a nástrojů pro plnohodnotný chod zařízení. Raspbian poskytuje více než 35 000 předkompilovaných balíčků. [4]



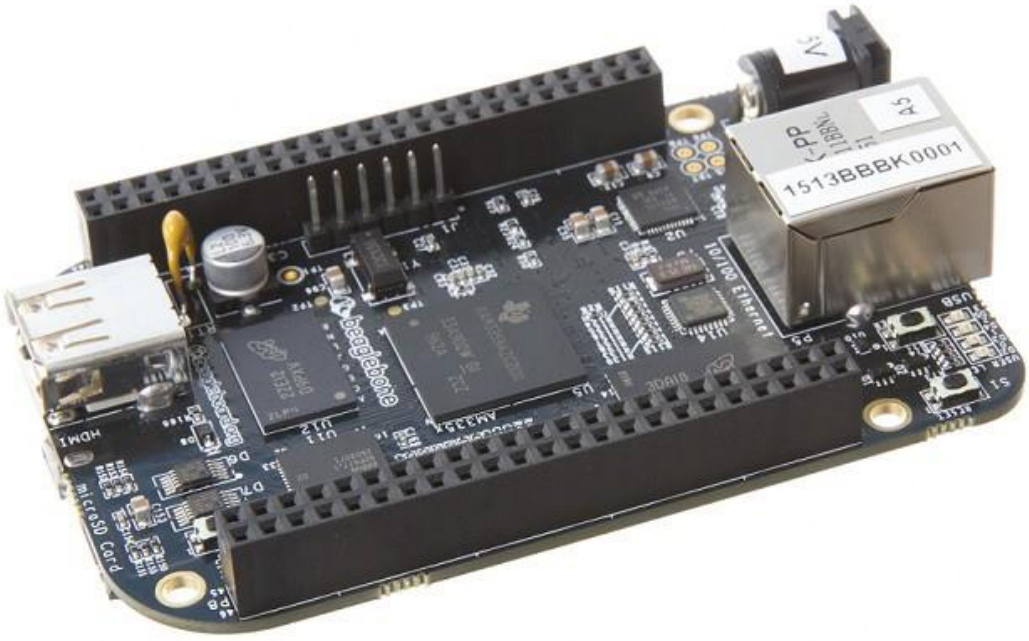
Obrázek 2: Raspbian [4]

1.2 BeagleBone Black

Zařízení s procesorem AM3358 Sitara od Texas Instruments poskytne vývojářům snadný přístup k průmyslovým standardním rozhraním a množstvím programů a nástrojů v oficiální distribuci. Srdce desky vychází z ARM procesoru Cortex-A8 doplněného o subsystém programovatelných jednotek v reálném čase a subsystém průmyslové komunikace (PRU-CSS).

Deska je osazena integrovaným obvodovým čipem TPS65217C pro řízení spotřeby a regulátorem pro práci s nízkým napětím. Operační paměť je zde o velikosti 512 MB typu DDR3 a vnitřní uložiště eMMC o velikosti 4 GB. Dostupný slot na microSD karty umožňuje běh systému z media, nebo k nahrání obrazu do vnitřního uložiště.

Z rozhraní jsou zde zastoupeny konektory USB 2.0, Ethernet 10/100 RJ45 a seriál port UART0, který je dostupný přes 6 pinů na desce. Zařízení podporuje rozšíření pomocí modulů. Pro konfiguraci a ladění Android zařízení lze použít zařízení jako Open Accessory Kit. [5]



Obrázek 3: BeagleBone Black [2]

1.2.1 Angstrom / Debian

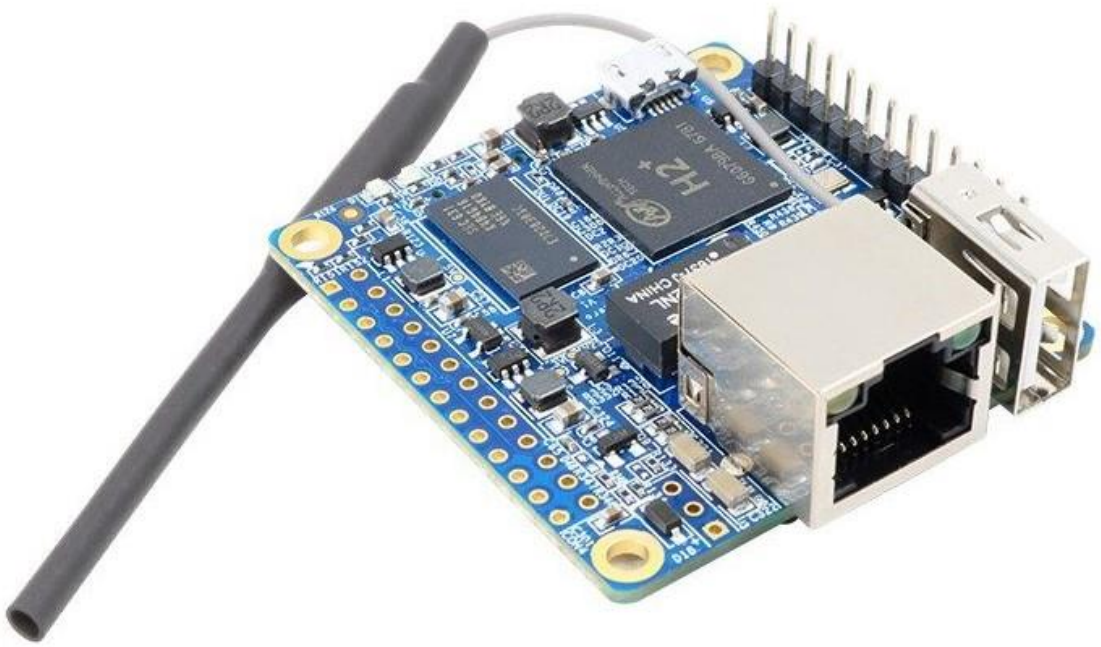
Oficiálně podporovaný systém není dnes již udržován a výrobce zařízení odkazuje na rozšířenou verzi Debian o BeagleBone software. Umístění Linuxu je u tohoto zařízení možné na microSD kartě, nebo na interním uložišti eMMC. [5]

1.3 Orange Pi Zero

Nejmenší z testovaných zařízení, které nedisponuje grafickým výstupem, je Orange Pi Zero. Pro přístup do systému je nutné použít sériový port, nebo síťové rozhraní.

Deska je osazena běžně používaným hardwarem, 32bitovým ARM procesorem Cortex-A7 od výrobce Allwinner se čtyřmi jádry a grafickou jednotkou Mali400MP2. Zařízení je osazeno sdílenou operační pamětí 256 MB.

Zařízení disponuje WiFi modulem XR819 802.11 s podporou standardu b/g/n a Ethernet RJ45 s možností napájení (POE). Dostupný je pouze jeden slot pro USB. Nízkoúrovňové rozhraní nabízí celkem 39 pinů. Použitím pinů se zpřístupní další dvě USB rozhraní. Ostatní piny jsou určeny pro připojení IR přijímače a zpracování zvuku. [6]



Obrázek 4: Orange Pi Zero [6]

1.4 Vzájemné porovnání

Pro vzájemné porovnání byla použita tři zařízení lišící se konstrukcí a uplatněním. Nejvíce známé Raspberry Pi 3 je průkopníkem sázejícím na držení kroku s aktuálními trendy a integruje mnoho prvků pro nejrozšířenější možnosti použití. Zařízení má pravidelné aktualizace softwaru a nejsnadněji dohledatelné volně dostupné návody a rady.

Zařízení s bezpečnostními prvky a vybavením pro využití v průmyslu charakterizuje BeagleBone Black. Vnitřní uložiště umožňuje použit volný microSD slot pro ukládání dat, nebo nabízí k dispozici druhý systém dosažitelný pouhým restartováním zařízení.

Nejkompaktnější a nejlehčí ze zvolených zařízení je Orange Pi Zero. Společně s možností napájení po ethernetovém kabelu se zdá být nejvhodnějším kandidátem do prostor, kam se nevejdou konkurenční výrobky.

2 Základní prvky Linuxu

Pod označením Linux si dnes mnozí představí nejen svobodný a otevřený, tedy s dostupným zdrojovým kódem, základ operačního systému, ale i ovladače, grafické prostředí a nespočet nainstalovaných aplikací, které posouvají možnosti systému daleko za rámec základního použití.

Rozšířený, či upravený obraz Linuxu je označován termínem distribuce. Výběr distribuce je vhodné zvolit podle účelu použití a zkušenosti uživatele.

RISC Linux distribuce, tedy operační systémy Linux určené pro procesory s redukovanou instrukční sadou, jsou dnes nejčastěji označované jako ***ARM Linux*** distribuce podle firmy ARM Holdings, která s architekturou přišla, a mají tak největší podíl zastoupení na trhu. [16]

2.1 Kernel

Jádro operačního systému Linux zprostředkovává přístup programů k hardware zařízení. Při spuštění počítače je zavedeno a drženo v operační paměti. Paměť je uvolněna až s vypnutím systému.

Zavedené jádro zajišťuje správu prostředků, umožnuje spouštění programů a zprostředkovává systémové služby.

2.2 Moduly

Moduly umožňují za běhu systému rozšířit funkcionalitu jádra. Nevyužívaný modul je možné odpojit a tím zachovat jen nezbytně nutnou výchozí velikost jádra, která je načtena v operační paměti.

Objektový soubor, který se za běhu nahraje do jádra operačního systému a rozšíří tak jeho funkcionalitu, je při ukončení jeho používání vyjmut a operační paměti uvolněna.

Typickým použitím modulů jsou ovladače zařízení, které jsou nahrány při připojení a odebrány po odpojení zařízení.

3 Toolchain

Toolchain je svazek základních developerských nástrojů, kterými jsou **kompilátor**, **binutils** a **glibc**. Svazek bývá rozšířený o nástroj pro ladění programu.

Toolchain používaný pro vývoj na odlišnou architekturu je označován jako „***cross compiler***“. Umožňuje běh veškerých nástrojů na hostitelském systému a generování spustitelných souborů pro architekturu cílovou. V našem případě bude náš hostitelský systém běžné PC na architektuře x86-64 a cílená platforma SoC s RISC procesorem. [17]

3.1.1 Binutils

Binutils je sada programovacích nástrojů pro vytváření a správu binárních programů, objektových souborů, knihoven a zdrojového kódu ve strojovém jazyce. [8]

3.1.1.1 Assembler

Jazyk symbolických adres, ***Assembly language***, je nízkoúrovňový jazyk se specializací podle architektury. Nativní zástupce jazyka pro Linux je *Gas*, jako nízkoúrovňová část překladače GCC. Vyšší programovací jazyky jsou přeloženy do jazyku symbolických adres, ve kterém už odpovídají strojovým instrukcím. Nástroj Assembler následně převede jazyk symbolických adres do strojového kódu. [1]

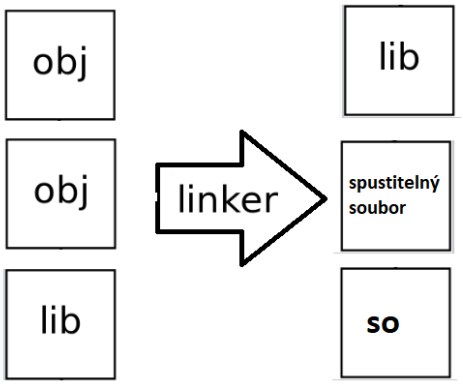
```
my_value    dd      12345678h       ; Proměnná o velikosti 32 bitů

    mov     eax, 1                  ; Naplní registr EAX hodnotou 1
    mov     ebx, eax                ; Zkopíruje obsah registru EAX do registru EBX
    mov     ecx, dword ptr dwValue  ; Naplní obsah registru ECX hodnotou uloženou v proměnné my_value
```

Obrázek 5: Ukázka jazyka Assembler

3.1.1.2 Linker

Linker je používaný nástroj v unixových systémech a je krátce označován ***ld***. Program slouží ke spojení vygenerovaných kódových objektů pro vytvoření spustitelného souboru. [1]

Binutils také obsahuje ***profiler*** pro výkonnostní analýzy aplikací, nástroj pro práci s archivy, utilitu pro zjištění velikosti, a další.



Obrázek: 6: Princip linkeru

3.1.2 Knihovna C

Knihovna jazyka C, zkratkou označována **libc**, je standardní knihovna pro programovací jazyk C, specifikována v *American National Standards Institute* a nazývána **ISO C library**. Klíčové vlastnosti jsou vysoká efektivita a přenositelnost. Projekt byl zahájen kolem roku 1988 a aktualizace jsou implementovány se zpětnou kompatibilitou.

Implementace pro unix systémy je ***GNU C Library,*** známá pod pojmem **glibc**. Je rozšířená o podporu nových standardů, jako je *ISO C11*, *POSIX* a dalších. Zahrnuje základní API (*open, read, write, malloc, ...)* a další nedílné prvky pro vývoj a práci v unixových systémech.

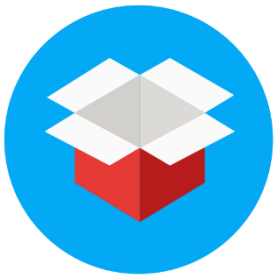
Pro procesory s omezenou instrukční sadou vznikly odlehčené **libc** knihovny, které by měly splňovat stejnou funckionalitu, například **uClib-ng**. [7]

3.2 Init

Init čili inicializace, je první puštěný proces během bootování systému, kterému je přiřazen identifikátor **1** a plní funkci předka všem ostatním procesům. Proces se spouští jako proces na pozadí – *deamon*, a běží po celou dobu trvání systému. Vzniká a zaniká společně s jádrem operačního systému v paměti. Mezi nejčastější init programy patří **SystemV**, **systemd** a pro embedded zařízení **BusyBox.** [1]

3.2.1 BusyBox

Základní sadou nástrojů cílených do embedded zařízení je BusyBox. Robustnější předloha SystemV, která je převážně používána v desktop Linux distribucí, je tímto řešením nahrazena a slibuje poskytnout stejně plnohodnotné nástroje. Přehled několika používaných. [9] [10]



Obrázek 7: BusyBox ikona [10]

```
init //definice pravidel programů při bootování a ukončování systému
ash // shell
cat // vypsání obsahu souboru
chmod // nastavení práv souborů a složek
cp // copy – nástroj pro kopírování
mv // move – přesunutí souborů
rm // remove – odstranění souboru
date // výpis datumu a času
dd // vytvoření bitové kopie
df // paměťové informace o systému
dmesg // info kernel bufferu
echo // poslání textu
grep // filtr výstupních data pomocí regulérních výrazů
gzip // aplikační software pro kompresy dat
kill // systémové ukončení procesu
mkdir // vytvoření adresáře
vi // textový editor
```

3.2.1.1 utilita dd

Nástroj pro nízkoúrovňové kopírování.

I. Benchmark testu disku a sekvenční analýza systémového výkonu

```
dd if=/dev/zero bs=1024 count=2000000 of=file_2GB // čtení
dd if=file_2GB of=/dev/null bs=1024 // zápis
```

II. Generování souboru s náhodnými daty

```
dd if=/dev/urandom of=myrandom bs=1000 count=1 // 1000bajtový soubor
```

III. Pro nahrání obrazu disku se systémem na paměťové medium.

```
sudo dd bs=4M if=image.img of=/dev/sdX conv=fsync
```

3.3 Bootloader

Po zapnutí počítače proběhne kontrola a nastavení základních prvků hardwaru, následně bootloader spustí zavaděč operačního systému.

3.3.1 Barebox

Zavaděč primárně používaný v embedded zařízeních je Barebox. Vychází ze zavaděče ***U-boot*** a získat lze volně z oficiálních stránek https://www.barebox.org/.

```
flex // lexikální analyzátor
bison // lexikální analyzátor
arm-linux-gnueabi-gcc // cross kompilér pro ARM
```

Barebox disponuje také rozhraním konfiguračního průvodce používajícího **KConfig**. Pro možnou vlastní konfiguraci jsou zde k dispozici uživatelská rozhraní **menuconfig** a **xconfig**, která budou popsána v kapitole Builroot.

Součástí staženého balíčku Barebox jsou předdefinované konfigurační soubory pro mnoho zařízení. Před kompilací zavaděče je nutné přednastavit prostředí dle typu desky.

```
export ARCH=arm // architektura ARM
export CROSS_COMPILE=/usr/bin/arm-linux-gnueabi-
// cesta k příslušnému kompilátoru, Makefile si ve výchozím nastavení doplnil „gcc“ na konci cesty sám
make rpi_defconfig // nastaví pracovní konfigurátor .config dle šablony pro dané zařízení
make menuconfig // uživatelská úprava konfigurátoru pro zavaděč
make // kompilace
```

Takto zkompilovaný zaváděcí soubor a veškerý obsah z adresáře **boot/** musí být nakopírovaný na první oddíl paměťového media embedded zařízení. Systém bude umístěný na jiném oddíle. Na prvním oddíle také bude konfigurační soubor, v kterém budou parametry pro spuštění jádra, jako je frekvence procesoru, povolené výstupy, vyhrazená paměť pro grafické jádro a mnohem více v závislosti na typu zařízení.

Na prvním oddílu je standardně **FAT** souborový systém o velikosti řádově jen několik MB. Zaváděcí oddíl je vždy nutné sestavit pro každé zařízení individuálně. V rámci použití frameworků je zavaděč vytvořen během procesu generování systému. [11]

3.4 Framework

Jednoduchý, efektivní a snadno použitelný nástroj ke generování embedded Linux systémů napříč architekturami. Důležitým prvkem je zachování jednoduché struktury umožňující snadné pochopení a jednoduchost vlastního rozšíření.

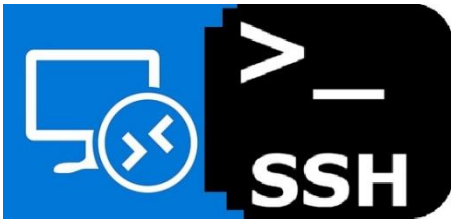
Framework umožňuje nastavit jednotlivé části operačního systému pomocí předpřipravené nabídky alternativ pro jednotlivé části. Od základních stavebních prvků až po doplňkové programy.

4 Pracovní stanice

Na hostitelské stanici byl nainstalován systém Debian 10 Buster ve výchozí konfiguraci. Jedná se o serverovou konfiguraci počítače, ke které je možné se připojit i vzdáleně.

V systému je nainstalován **SSH server** a povolen vzdálený přístup.

Obrázek 8: Logo Debian



Obrázek 9: RDP / SSH

Pro grafický přístup z operačního systému z dílen Microsoft je v systému nainstalována podpora pro *Remote Desktop Protokol*.

```
sudo apt install xrdp
sudo systemctl enable xrdp // automatické spuštění po náběhu OS
```

Veškeré příkazy jsou zadávány do textového uživatelského rozhraní **shell**. Pro daný operační systém je výchozí *shell* zvolen **bash**.

4.1 Vliv paměťového media na výkon systému

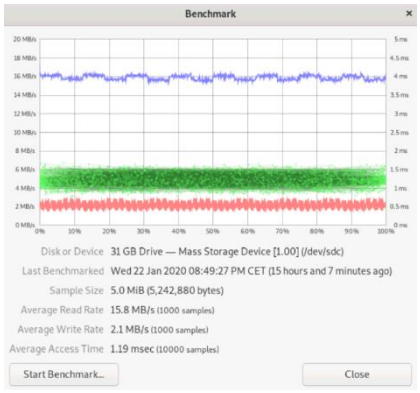
Pro účely testování byly k dispozici tři paměťové karty s velikostí 16 GB a dvakrát 32 GB rychlostní třídy deset. Pro porovnání jednotlivých karet byl použit nástroj ***gnome disk***. Paměťová media bývají zpravidla nejpomalejší prvky systému. V případě využití SD karet se tvrzení potvrdilo.

```
sudo apt install gnome-disk-utility
sudo gnome-disks // pro spuštění
```

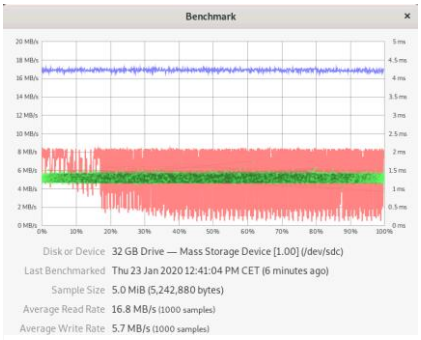
Obrázek 10: SD karta 16 GB benchmark

První testovací paměťové medium značky Kingstom o velikosti 16 GB vykazovalo stabilní rychlost čtení okolo 15.8 MB/s. Průměrná přístupová doba se pohybovala nad 1 msec. Rychlost zápisu kolísala v poměrně širokém rozsahu a průměrná hodnota se stanovila na 3.3 MB/s.

Druhá testovací SD karta stejné značky Kingstom, o udávané velikosti 32 GB, disponovala ustálenými hodnotami. Rychlost čtení byla ustanovena na 15.8 MB/s. Vykazovala nejstabilnější a nejpomalejší zápis s průměrnou hodnotou 2.1 MiB/s. Přístupová doba byla průměrně 1.19 msec.



Obrázek 11: SD karta A 32 GB benchmark



Obrázek 12: SD karta R 32 GB benchmark

Třetí karta stejné značky o velikosti 32 GB vykazovala opět rozdílnou charakteristiku. Průměrná rychlost čtení se ustálila na 16.8 MB/s. Rychlost zápisu se v průměru ustálila na 5.8 MB/s a doba přístupu na 1.27 msec.

Podle námi zvoleného testovacího profilu tato karta udává nejlepší hodnoty pro čtení a zápis, ale v přístupové době se umístila až na poslední příčce.

Vliv výkonu systému na paměťové kartě se projeví zejména při bootování operačního systému a spouštění aplikací.

5 Buildroot

Obrázek 13: Buildroot [3]

Buildroot je nástroj pro zjednodušení a automatizaci sestavení kompletního Linuxového operačního systému pro embedded systémy pomocí cross-compilace. Jedná se tedy o nástroj, který umožňuje kompilaci **toolchain**, **root filesystemu**, Linux **kernelu** a **bootloader** pro odlišnou architekturu.

Standardní průběh kompilace Buildrootu je **stažení balíčku**, **rozbalení**, **konfigurace**, **kompilace** a **instalace**, tedy umístění na paměťové medium. Zdrojový kód je rozbalen do ***output/build/<package>-<version>*** jako dočasný. Při **make clean**, nebo opětovném **make** je promazán a znovu vytvořen. [3]

5.1 Požadavky na systém

Buildroot, jakožto nástroj pro sestavení linuxového operačního systému, je čistě linuxová aplikace. Pro chod frameworku je nutné mít v systému kromě běžně používaných programů i několik specifických, které bude nutné doinstalovat. Některé, zejména ze sekce *Volitelné programy*, jsou využity jen při určitých specifikacích. [3]

Požadované programy nainstalujeme pomocí balíčkového systému, v našem případě **dpkg.**

```
apt install název-požadovaného-programu // komentar
```

5.1.1 Požadované programy

```
build-essential // jen pro systém Debian
which // zobrazí cestu spustitelného programu
sed // pro vyhledávání a nahrazování stream textu
make // utilita pro automatizaci překladu
gcc, g++ // překladač, sada překladačů (verze 4.8+)
bash // uživatelské rozhrání
patch // program k aplikování změn, používá se s programem diff
gzip, bzip2, unzip // pro kompresy a baleni dat
perl // interpretovaný skriptovací jazyk (verze 5.8.7+)
tar, cpio // pro práci s archivy
rsync // synchronizace souborů a adresářů
file // pro zjišťování typu souborů (umístění musí být v /usr/bin/file)
bc // pro numerické výpočty
wget // stahovač souborů
```

5.1.1.1 Doplňkové programy

Programy, které nejsou uvedeny v dokumentaci, ale v průběhu konfigurace systému byly vyžádány.

```
pkg-config // informace o knihovnách pro účely vlastního překladu
python // vysokoúrovňový skriptovací jazyk (verze 2.7+)
python-pip // správce balíčků Pythonu
```

5.1.1.2 Stahovače zdrojů

Kromě základního nástroje **wget,** který je uveden v kapitole 4.1.1, je několik balíčků, které jsou dostupné pouze prostřednictvím verzovacích systémů. Pro možnost stažení Buildrootem je nutné doinstalovat několik dalších nástrojů. [3]

```
bazaar // Bazaar-NG
cvs // Concurrent Versions System
git // distribuovaný systém správy verzí vytvořený Linusem Torvaldsem
mercurial // obdoba gitu
rsync // slouží k synchronizaci obsahu souborů a adresářů
scp // zabezpečený protokol pro přenos souborů po síti
subversion // TortoiseSVN je rozhraní / nástavba
```

5.1.1.3 Java rozšíření

V případě úmyslu zaintegrovat do skládaného systému Javu, je nutné hostitelský systém rozšířit i o tyto balíčky. [3]

```
asciidoc // úsporný značkovací jazyk (verzi 8.6.3+)
w3m // webový prohlížeč pracující v textovém uživatelském rozhraní
argparse // našeptávač pro Python
dblatex // LaTeX editor (pro PDF manuál)
```

5.1.1.4 Podpora pro grafická znázornění

Pro grafy a grafické znázornění závislostí jsou potřeba tyto nástroje. [3]

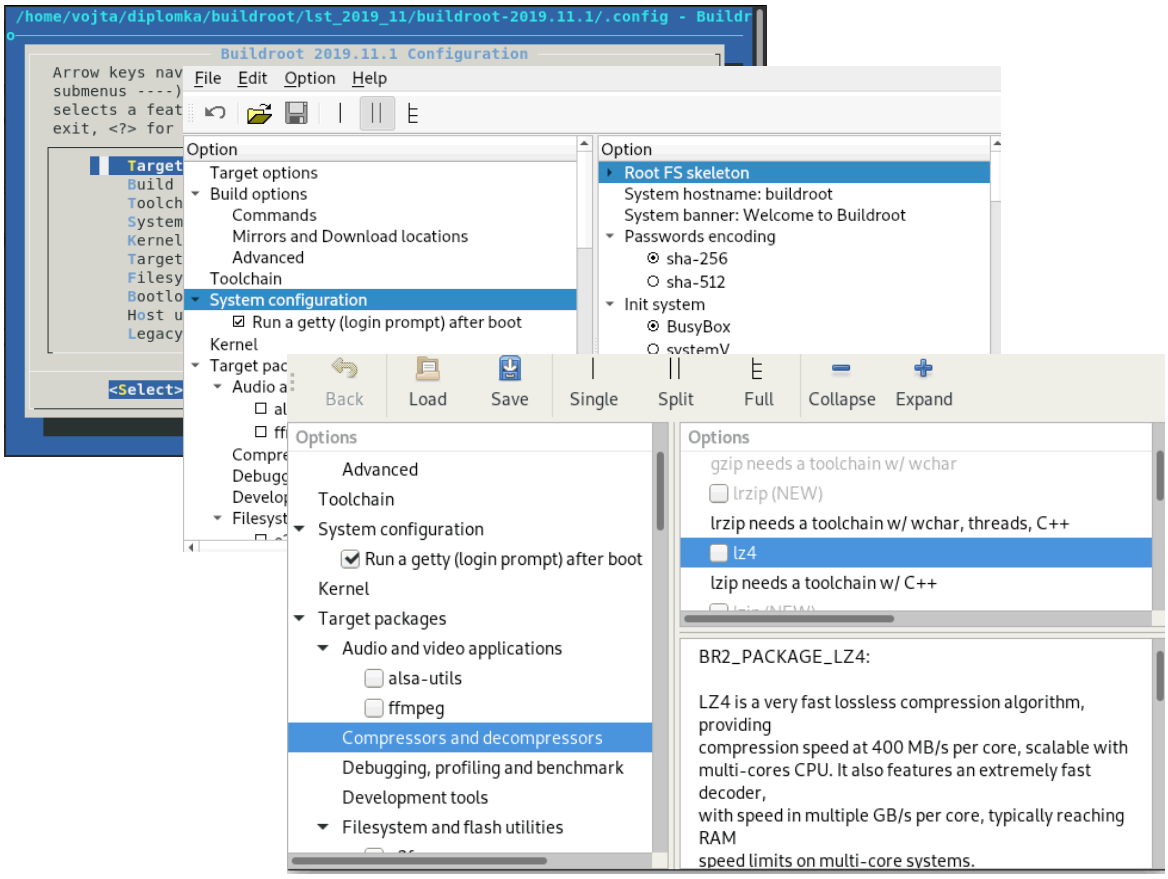
```
graphviz // pro graph-depends a <pkg>-graph-depends
python-matplotlib, graph-numpy // pro graph-build
```

5.1.2 Výběr konfiguračního rozhraní

Pro odpůrce textového rozhraní jsou v nabídce přidány grafické alternativy zefektivňující využití robustního systému. Pro tuto volbu je nutné mít v systému zprovozněné grafické rozhraní. Runtime verze je potřeba rozšířit i o vývojářské verze. Názvy developerských verzí jsou často rozšířeny o koncovku ***-dev***, nebo ***-devel***. [3]

```
config // textový výpis, ne scela vhodný pro rozsáhlost
ncurses5 (-dev) // pro rozhraní menuconfig
qt5 (-dev) // pro rozhraní xconfig
glib2, gtk2 a glade2 // pro rozhraní gconfig
```

Rozhraní **menuconfig** běží v textovém režimu. Ostatní rozhraní jsou grafická a vyžadují grafické knihovny a nástroje. [13]



Obrázek 14: Rozhraní menuconfig / xconfig /gconfig

5.2 Získaní frameworku Buildroot

Na stránkách projektu https://buildroot.org/ je několik možností, jak nástroj stáhnout. Kromě možnosti stažení souboru přes webové rozhraní v kompresních baličkách **tar.gz**, a **tar.bz2** je doporučená alternativa pomocí nástroje **git**.

```
git clone git://git.buildroot.net/buildroot
```

5.2.1.1 Vagrant

Buildroot je dostupný také přes nástroj Vagrant. Pomocí tohoto nástroje se jediným příkazem stáhne obraz operačního systému předpřipraveným Buildrootem pro okamžité spuštění konfigurátoru. Předpokladem je mít stažený samotný **Vagrant**, virtualizační nástroj, například **VirtualBox,** a hardware podporující virtualizaci. [3]

5.3 Konfigurace projektu

Průvodcem výběru jednotlivých komponent celého operačního systému pro následnou kompilaci je konfigurátor, který nabízí až čtyři uživatelská rozhraní, z čehož tři používají grafické komponenty a jeden terminálový způsob tvorby rozhraní.

Název konfiguračního souboru pro kompilační proces je pevně nastaven na **.config**. Pro uložení konfiguračního souboru je vyhrazen adresář **configs**, kde je vhodné soubor ukládat podle klíčových charakteristik. Konec názvu musí obsahovat ***_defconfig***. [3]

```
rspi3_uglibc_minimalist_defconfig // pi3_arch_toolchain_minimální
```

Do pracovního konfiguračního souboru ***.config*** se ukládají veškeré změny při uložení konfiguračního průvodce. Uložení do adresáře se provede příkazem.

```
make savedefconfig
```

Název ukládaného konfiguračního souboru se specifikuje v konfigurátoru pod položkou ***Build Options***.

Nahrání konfiguračního souboru z úložného adresáře do pracovního konfiguračního souboru ***.config*** se provádí příkazem ***make*** a názvem požadovaného souboru. [3]

```
make rsppi3_uglibc_minimalist_defconfig
 => #configuration written to .config
```

5.3.1 Příkaz make

Po nakonfigurování a uložení konfiguračního souboru je spuštěn proces sestavování operačního systému příkazem **make**.

```
make // spuštění procesu generování vlastního operačního systému
```

Příkaz **make** provede následující kroky:

1. stáhne požadované / chybějící soubory
2. nakonfiguruje, sestaví a nainstaluje cross-kompilační nástroje
3. nakonfiguruje, sestaví a nainstaluje vybrané balíčky
4. sestaví kernel
5. sestaví obraz zavaděče
6. vytvoří root filesystem

Výstup příkazu je uložen do adresáře **output/** do jednotlivých složek.

```
images/ obrazy kernelu, zavaděč a filesystem, pro cílové zařízení
build/ komponenty, které Buildroot potřebuje
host/ nástroje pro host PC a sysroot cílového toolchainu
staging/ symlink na host/, kvůli zachování kompatibility
target/ téměř celý root filesystem pro cílové zařízení
```

Při změně ***architektury***, nebo ***toolchain*** konfigurace je potřeba promazat předchozí kompilace. Před zahájením pro jiné cílové zařízení je nutné promazat i konfigurační soubor.

```
make clean // promazání zkompilovaných balíčků
make distclean // promazání včetně konfiguračních souborů
```

Vygenerované manuálové stránky budou k dispozici ve složce **/manual**.

```
make manual(-clean) // vygenerování manuálových stránek (promazání)
```

V manuálových stránkách je popsáno, kdy je potřeba udělat nový čistý build. Pro zjednodušení problematiky je zde dostupná varianta příkazu, která provede novou čistou kompilaci, tedy vše zkompiluje znovu. [3]

```
make clean all // promazání zkompilovaných balíčků a opětná kompilace
```

Samotné stažení souborů pro možnou pozdější kompilaci bez připojení k internetu je provedeno parametrem ***source***.

```
make source // stažení balíčků pro offline kompilaci
```

Pro kompilaci mimo výchozí cestu **output/** je syntaxe následující.

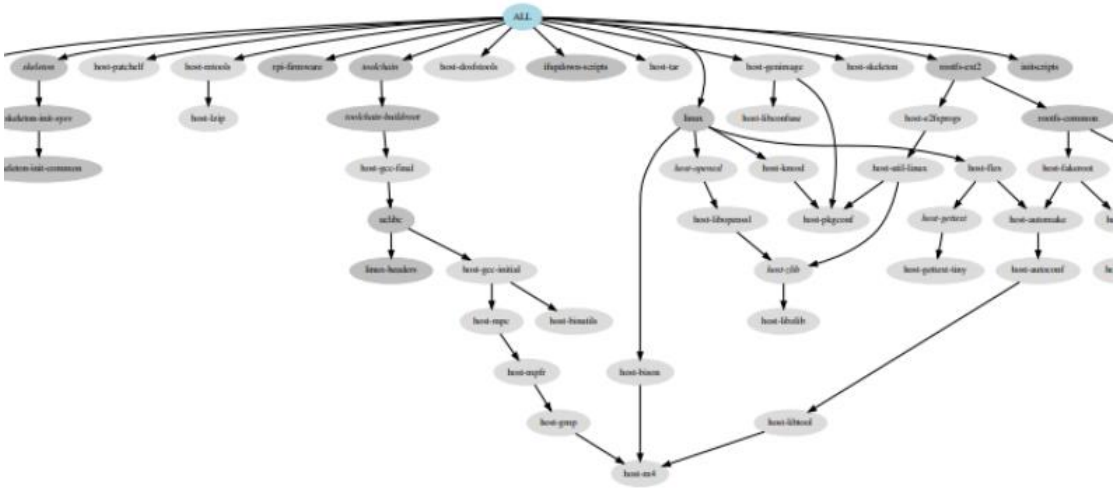
```
make O=/custom/abs/path // kompilace proběhne do /custom/abs/path
cd /custom/abs/path; make O=$PWD -C path/to/buildroot // alternativa
make /custom/abs/path // pro opakovaní procesu
```

5.3.1.1 Grafické zobrazení

Na systémech disponující grafickými knihovnami nabízí Buildroot grafické znázornění. Veškeré výstupy budou k nalezení v adresáři **output/graph** ve formátu PDF.

Jedna z neocenitelných funkcí zobrazující závislosti mezi jednotlivými balíčky je vyvolána parametrem **graph-depends**. [3]

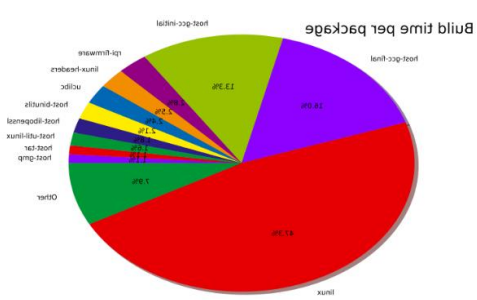
```
make graph-depends // graf závislostí balíčků [rdepends]
```



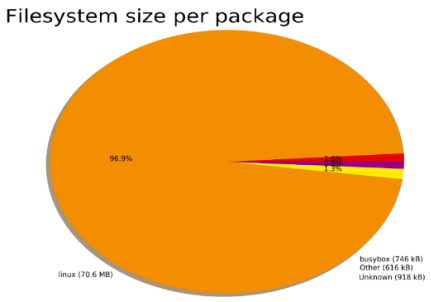
Obrázek 15: Graph – depends

Pro přehled doby časové náročnosti je implementován příkaz **graph-build** a paměťové rozložení příkazem **graph-size**.

```
make graph-build, make graph-size
```



Obrázek 17: Graph – build

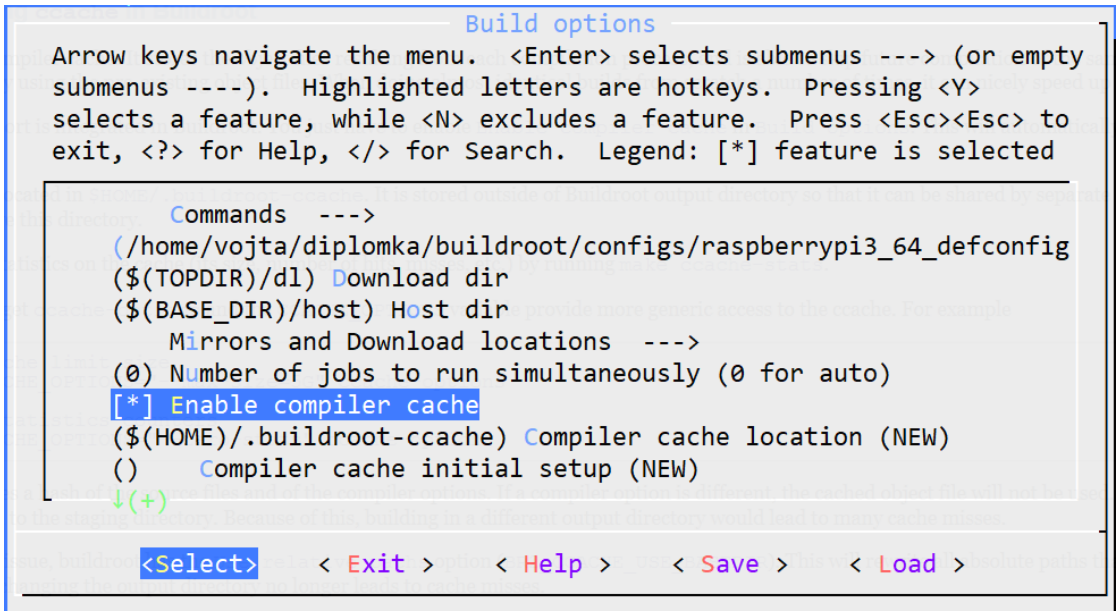


Obrázek 16: Graph – size

Veškeré příkazy je doporučené provádět pod běžným uživatelem. Předchází se tak případnému poškození vlastního operačního systému. [3]

5.3.2 Mezipaměť mezi kompilacemi

Mezipaměť kompilátoru je označena ***ccache***. Ukládá výsledné soubory objektů z kompilace pro možné budoucí opětovné užití. Povolení / zakázání je přístupné ze samotného průvodce konfigurace. [3]



Obrázek 18: Konfigurátor – cache

Výchozí cesta pro uložení mezipaměti je **$HOME/.buildroot-cache**, tedy mimo adresář Buildroota z důvodu možnosti sdílení mezi více projekty. Cestu lze změnit.

```
make CCACHE_OPTIONS="–max-size=15G" ccache-options // max. velikost
```

5.3.3 Proměnné prostředí

Průběh procesu lze ovlivnit pomocí proměnných, nebo rozšiřujících konfiguračních souborů. Pro běžné použití není potřeba tato rozšíření použít. Veškeré základní nutné nastavení lze provést i přes výchozí konfigurační průvodce dostupné příkazem ***make menuconfig.*** [3]

```
HOSTCXX // explicitně dosazený C++ kompilér
HOSTCC // explicitně dosazený C kompilér
UCLIBC_CONFIG_FILE=<path/to/.config> // rozšířená uClibc konfigurace
BUSYBOX_CONFIG_FILE=<path/to/.config> // rozšířená BusyBox konf.
BR2_CCACHE_DIR // rozšíření konfiguračního souboru CACHE
BR2_DL_DIR // rozšíření konfiguračního souboru DOWNLOAD
BR2_GRAPH_ALT, BR2_GRAPH_OUT, BR2_GRAPH_DEPS_OPTS, BR2_GRAPH_DOT_OPTS,
BR2_GRAPH_SIZE_OPTS // rozšíření grafických konf. souborů
```

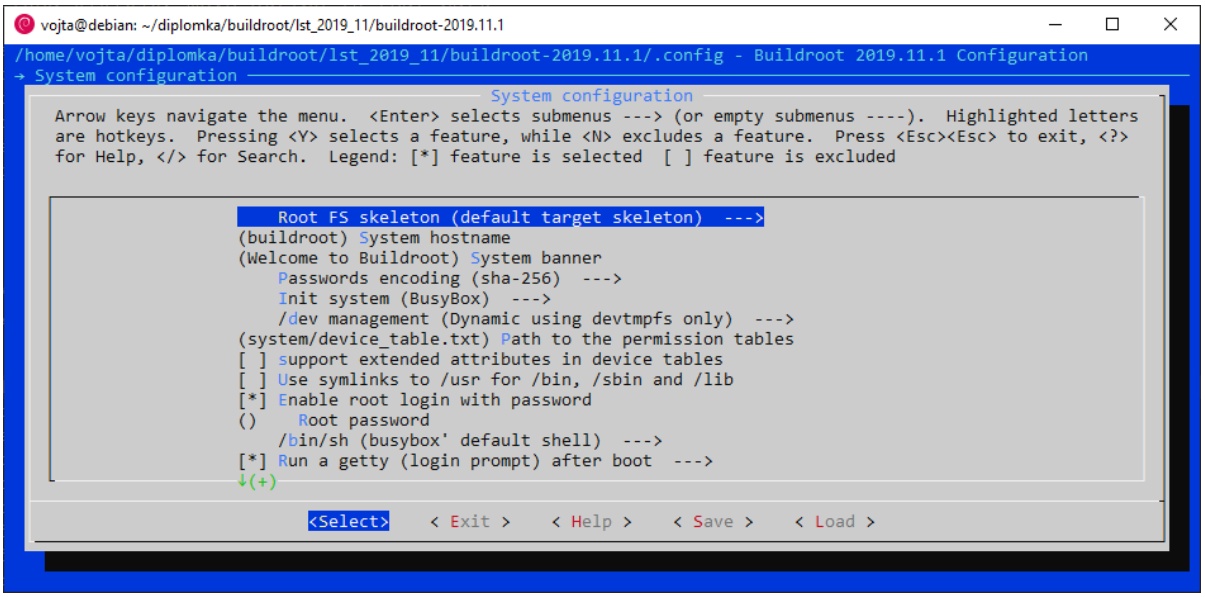
Způsob použití rozšíření.

```
make UCLIBC_CONFIG_FILE=uClibc.cfg BUSYBOX_CONFIG_FILE=$HOME/bb.cfg
// externí uclibc a busybox konfigurační soubor
make HOSTCXX=g++-4.3-HEAD HOSTCC=gcc-4.3-HEAD
// explicitně dosazené překladače
```

5.4 Konfigurační průvodce

Po instalaci veškerých nutných programů uživatel spustí průvodce konfigurace vlastního operačního systému. Z praktických důvodů není základní textový konfigurátor ***config*** doporučený. Základní uživatelské rozhraní v textovém režimu běžící v terminálu je ***menuconfig***. [3]

```
make menuconfig // nconfig, xconfig, nebo gconfig pro výběr grafických rozhraní
```



Obrázek 19: Make menuconfig

5.4.1 Rozšířené možnosti konfigurace

Pro samotnou, nebo rozšířenou konfiguraci jednotlivých částí, je do Builroota implementována možnost spuštění samostatně. Krom ***menuconfig*** lze konfigurátor spustit i v ostatních variantách. [3]

```
make busybox-menuconfig // separátní konfigurace busybox
make linux-menuconfig a linux-savedefconfig // sep. konf. kernelu
make uclibc-menuconfig // separátní konfigurace uclibc
make uboot-menuconfig a uboot-savedefconfig // sep. konf. uboot
```

5.4.2 Práce nad balíčky – separátně

Při samotné změně některého z balíčku a vyhnutí se celé kompilace od nuly je možné provést kompilaci balíčku izolovaně.

```
make <package>-dirclean // smazání balíčku
make <package>-reinstall // nakopírování balíčků na určené místa
make <package>-rebuild // operace make a make install
make <package>-reconfigure // obnova konfigu., make a make install
```

V nabídce Buildroota je více možností, z nichž některé již byly zmíněné dříve. Pro zmínku stojí několik dalších, které se do předešlých kapitol nevešly.

```
make <package>-show-depends // alternativa show-recursive-depends
make <package>-source // samotné stažení zdrojů
make <package>-depends // stažení a instalace závislostí
```

Buildroot nehlídá závislosti, a je tedy nutné baličky s návazností překonfigurovat také. Informace o úpravách se nachází v cílovém adresáři ve skrytých souborech ***stamp_***. [3]

5.4.3 Použití run-testů frameworku

Buildroot obsahuje dvě rozdílné metodiky testování. První z nich využívá Python skripty k otestování správnosti konfigurace a druhá provádí spuštění sestaveného systému ve virtuálním stroji. [3]

5.4.3.1 Unittest

Kromě nutnosti nainstalování správce balíčků pro programovací jazyk Python, je nutné rozšířit vyhledávací cestu o lokální adresář.

```
python-pip // Python balíčkovací systém
pip install nose2 // získání nose2 pomocí Python správce balíčků
export PATH="$PATH:/home/$USR/.local/bin" // rozšíření cesty
```

Testování správnosti konfigurace pomocí skriptování obsahuje testy k otestování základních prvků a funkčnosti konfigurace jako celku.[3]

```
support/testing/run-tests -l // základní předpřipravený test
```

```
vojta@vojta:~/diplomka/buildroot$ support/testing/run-tests -l
List of tests
test_run (tests.package.test_lsqlite3.TestLuaLsqlite3) ... ok
test_run (tests.package.test_lsqlite3.TestLuajitLsqlite3) ... ok
test_run (tests.package.test_perl_dbd_mysql.TestPerlDBDmysql) ... ok
test_run (tests.package.test_python_pyyaml.TestPythonPy2Pyyaml) ... ok
test_run (tests.package.test_python_pyyaml.TestPythonPy3Pyyaml) ... ok
test_run (tests.package.test_python.TestPython2) ... ok
test_run (tests.package.test_python.TestPython3) ... ok
```

Obrázek 20: Unittest nose2

5.4.3.2 Qemu

Je zkratkou Quick EMUlátoru, který poskytuje hardwarovou a softwarovou virtualizaci. Stejně tak, jako ostatní virtualizační nástroje je i tento vhodný pro otestování funkčnosti, či využití vlastností daného hostovaného systému, ale z hlediska porovnání výkonosti je pro dané účely nepoužitelný. [12]

6 OpenADK

Obrázek 21: OpenADK avatar [2]

Nástroj pro sestavení systému pro embedded zařízení s cílem rychlého náběhu, úspornou velikostí a nativní bezpečností pro data a užití. V porovnání s Buildrootem je projekt méně rozmanitý a nabízí méně možností a variací. Proces se skládá z kompilace **toolchainu**, **root filesystemu**, Linux **kernelu** a **bootloaderu** pro cílové zařízení. [2]

Systém sestavený nástrojem je v režimu pouze pro čtení. Jediné trvale povolené změny jsou ve složce **etc/** a to jen v případě, že se před *rebootem* systému uloží pomocí nástroje ***cfgfs commit***. [2]

6.1 Získání frameworku OpenADK

OpenADK je volně dostupný na oficiálních stránkách projektu https://openadk.org/ prostřednictvím stažení skrze webové stránky, nebo stažení s možností pozdějšího aktualizování, či procházení kroky vývoje skrze **git**.

```
git clone git://openadk.org/git/openadk
```

6.2 Požadavky na systém

Pro spuštění je nutné unixový systém rozšířit o další programy a knihovny. Pro základní kompilaci stačí pouze základní rozšíření. V případě rozsáhlého projektu budou vyžadovány knihovny, které v dokumentaci nejsou uvedeny.

Požadované programy jsou instalovány pomocí balíčkovacího systému, v našem případě **dpkg.**

```
apt install název-požadovaného-programu // komentář
```

6.2.1 Požadované programy

Základní předpoklad pro průvodce a kompilaci jsou následující programy. Ve využívaném systému budou mnohé z nich již obsaženy. [2]

```
libc dev // knihovna jazyka C
binutils // standardní sada nástrojů používaných při programování
C, C++ // kompilátory
make // utilita pro automatizaci překladu
gzip, tar, zlib dev // archivace, komprese dat
perl // interpretovaný skriptovací jazyk (verze 5.8.7+)
git, curl, nebo wget // stahovače
ncurses dev // pro tvorbu rozhraní
```

6.2.2 Doplňkové programy

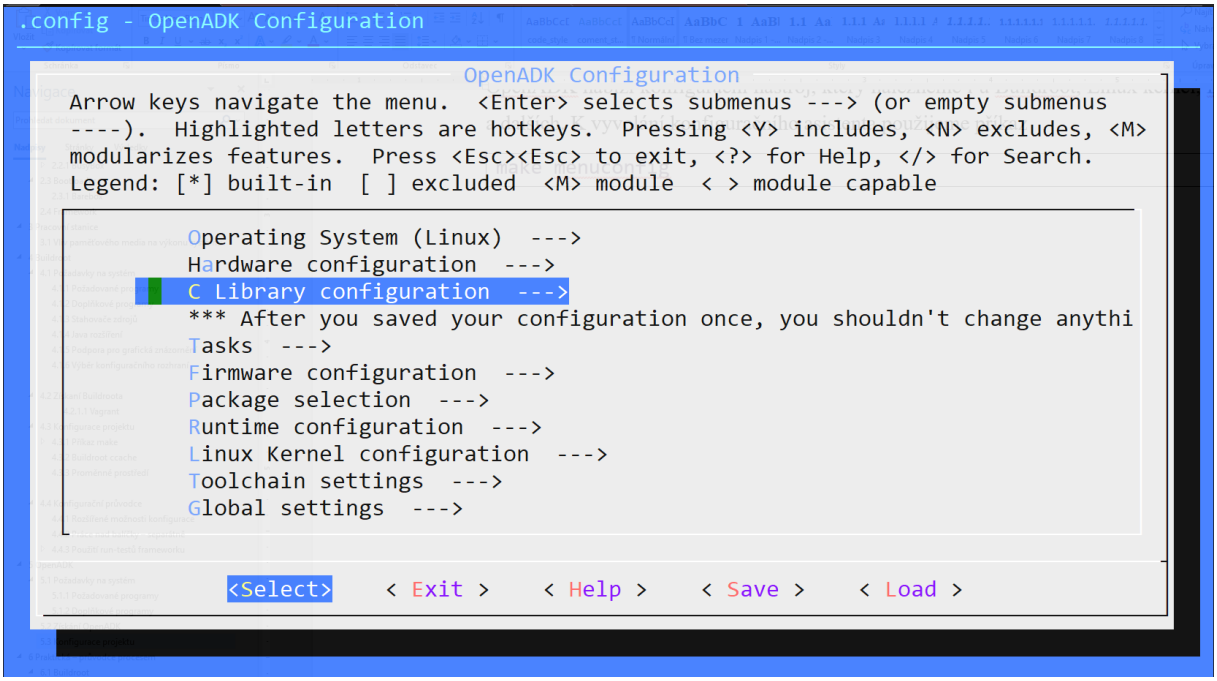
Programy a příslušné knihovny, které nejsou nezbytné pro základní kompilaci linuxového jádra. [2]

```
mksh // typ shellu
mesa // open source implementace OpenGL
ruby-full // interpretovaný skriprtovací programovací jazyk
```

6.3 Konfigurace projektu

OpenADK nabízí konfigurační nástroj, který se nachází i ve frameworku Buildroot, Linux kernel konfigurátoru a dalších. Konfiguračního asistenta vyvoláme příkazem **menuconfig**.

```
make menuconfig
```

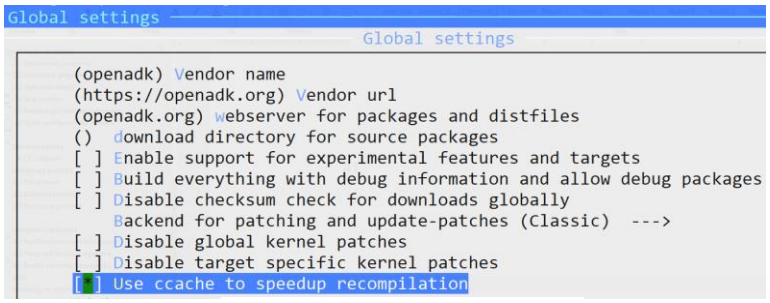


Obrázek 22: Make menuconfig

První část jsou pilíře systému, jako je architektura procesoru, typ desky a knihovna jazyka C. Veškeré další programy se kompilují z vybrané ***libc*** knihovny pro danou architekturu. Při změně těchto stavebních prvků je nutné provést kompletní rebuild veškerých programů a knihoven v systému. [2]

6.3.1 Mezipaměť mezi kompilacemi

Ccahe je mezipaměť kompilátoru mezi jednotlivými iteracemi vývoje. Ukládá výsledné soubory objektů kompilace pro opětovné využití. Nastavení použití je nastavováno v konfiguračním průvodci. [2]



Obrázek 23: OpenADK – cache

6.3.2 Příkaz make

Příkaz **make** se provede následujícími kroky:

1. stáhne zdrojové soubory
2. nakonfiguruje, sestaví a nainstaluje nástroje do hostitelského systému
3. nakonfiguruje, sestaví a nainstaluje cross-kompilační nástroje
4. sestaví obraz kernelu
5. sestaví a nainstaluje cílové balíčky
6. sestaví bootloader
7. vytvoří root file systém

Výstupy kroků procesu jsou pro přehled uloženy do separátních adresářů.

```
firmware/ // obrazy a balíčky
build_<system>_<libc>_<arch>_<abi>/ // komponenty
target_<system>_<libc>/ //instalačky a balíčky pro cílové zařízení
root_<system>_<libc/ // rootfilesystem cilového zaříení
host_<gnu_host_name>/ // kompilační nástroje
host_build_<gnu_host_name>/ // sestavené nástroje
toolchain_<system>_<libc>_<arch>_<abi>>/ // cross-compile nástroje
toolchain_build_<system>_<libc>_<arch>_<abi>/ // nástroje pro sestavení systému
pkg_<system>_<libc>_<arch>_<abi>/ // pro pkg
```

Při zásadních změnách konfiguračního souboru je nutné promazání předešlých kompilací. [2]

```
make cleandir // promazání pracovního adresáře
make distclean // promazání stažených souborů a konfigurace
make cleankernel // promazání kernelu
make clean // promazání balíčků
```

7 Praktická část – průvodce procesem vytvoření linuxového systému

Po instalaci hostitelského systému Debian je nutné rozšíření vyhledávací cesty spustitelných souborů.

```
PATH=$PATH:/usr/sbin // proměnná=původní+nová vyhledávací cesta
export PATH // aktualizace systémového prostředí
```

Pro administrátorská oprávnění běžného účtu bude používán nástroj **sudo**.

```
usermode -a -G sudo vojta // přidání uživatele do sudo skupiny
```

Pracovní adresář(e) byl umístěn do složky domovského adresáře.

```
$HOME/diplomka // pracovní adresář
```

7.1 Oficiálně podporované distribuce

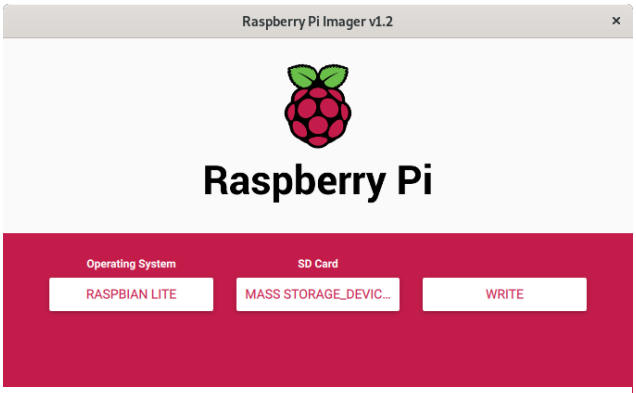
Doporučený způsob nahrání staženého systému do zařízení se liší, ale vždy se jedná o nakopírování vytvořeného obrazu na paměťové medium. Raspberry Pi má příjemně vyhlížející grafický nástroj nesoucí logo organizace. BeagleBone se odkazuje na použití univerzálního nástroje ***balenaEtcher*** a Orange Pi na svých stránkách popisuje linuxovou utilitu **dd**, která stejně tak zmíněné grafické nástroje provede bitovou kopii obrazu na SD kartu.

Oficiální systém pro Raspberry Pi je Raspbian, který je dostupný v desktopové verzi a ve verzi ***lite***, která neobsahuje grafické uživatelské rozhraní.

7.1.1 Raspbian Lite

Raspberry Pi imager má v sobě zakomponováno i stažení. Stačí pouze vybrat systém Raspbian lite, zvolit paměťové medium a proces potvrdit.

```
https://www.raspberrypi.org/downloads/ // Raspberry Pi imager
```



Obrázek 24: Raspberry Pi imager [4]

Výchozí uživatel Raspbianu je ***pi*** a heslo ***raspberry***.

```
pi / raspberry // výchozí uživatel / heslo
```

7.1.2 Beagle Debian

BeagleBoard na stránkách nabízí a doporučuje stažení upravené verze systému Debian, standardně bez grafického rozhraní.

```
https://beagleboard.org/latest-images // obraz systému Debian
```

Pro nahrání operačního systému je doporučeno použit grafický nástroj ***balenaEtcher***.

```
https://www.balena.io/etcher/ // univerzální flasher
```

Obraz systému Debian je zabalený v balíku ve formátu ***xz***, před nahráním je nutné jej rozbalit.

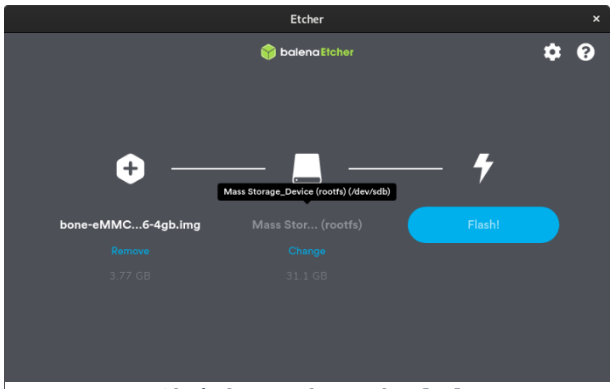
```
sudo apt-get install xz-utils // nástroj pro rozbalení
unxz bone-img.xz // rozbalení balíčku
```

BalenaEtcher byl spouštěn s parametrem ***–no-sandbox***.

```
./balenaEtcher-1.5.89-x64.AppImage -no-sandbox
```

Po výběru předem rozbaleného balíku bylo vybráno cílové medium pro nahrání systému.

Ke stažení jsou nabízeny dva systémy, z nichž je jeden určený k běhu z SD karty a druhý k nahrání do vnitřní paměti eMMC. Systémy jsou totožné a liší se pouze ve spouštěcím parametru.



Obrázek 25: BalenaEtcher [13]

```
/boot/uEnv.txt // cesta k souboru s parametry
// zakomentovaný řádek pomocí # = boot z SD karty
// odkomentovaný řádek = nahrání do vnitří paměti
cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh
```

Výchozí bootování je nastaveno na čtení z vnitřního uložiště. Pro čtení z SD karty z důvodu načtení systému, nebo nahrání do vnitřního uložiště, se provádějí následujícími kroky:

1. vložení media do zařízení
2. zmáčknutí a podržení tlačítka **USR/BOOT**
3. připojení napájecího kabelu

Zařízení je možné připojit k monitoru přes mikro HDMI konektor. Výchozí uživatel podle oficiálních zdrojů je „**debian**“ a „**root**“. [2]

```
debian / temppwd // uživatel / heslo
```

Logování za uživatele ***root*** není ve výchozím nastavení povoleno.

7.1.3 OrangePi Debian

V případě Orange Pi Zero jsou k dispozici tři varianty oficiálně podporovaných unixových systémů. Krom zvoleného systému Debian je zde dostupný systém Armbian, Ubuntu a Android.

Stažení oficiálně podporovaného systému je možné z oficiálních stránek v záložce ***download***.

```
http://www.orangepi.org/downloadresources/ // Debian
```

Pro nahrání obrazu systému na paměťové medium je doporučeno použít nástroj **dd**.

```
sudo dd bs=4M if=imagesd.img of=/dev/sdb conv=sync // unix utilita
```

Při použití univerzální grafické utility ***balenaEtcher*** je docíleno stejného výsledku.

Alternativní způsob přístupu do systému přes terminál je pomocí klienta SSH, který je nutné ve staženém obrazu disku před nahráním na paměťové medium povolit.

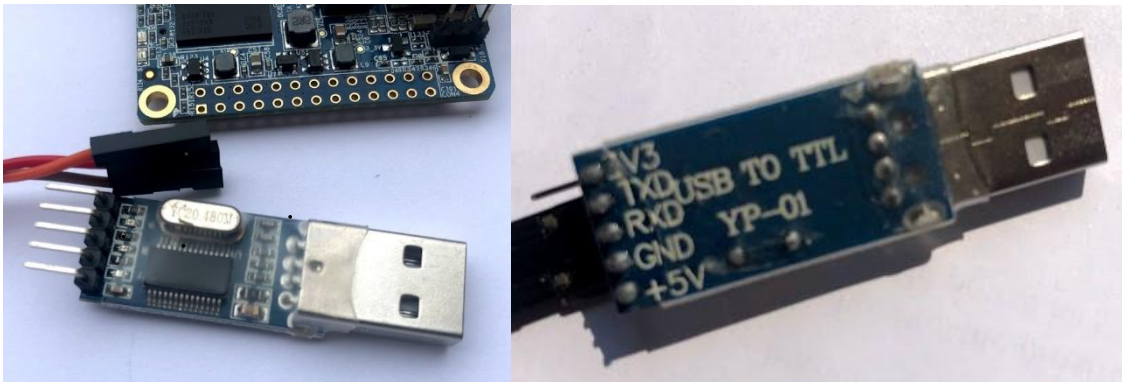
```
/etc/rc.local // vytvoříme soubor
// přidáme do něj řádek
/etc/init.d ssh start
```

Po připojení zařízení do sítě mu bude přidělena IP adresa – je nutné zjistit z routeru jaká IP mu mu byla přidělena. Následně je možné se do zařízení přihlásit pomocí SSH klienta.

```
ssh root@<ip_device>
orangepi // výchozí heslo k uživateli root
```

7.2 Vstup a výstup zařízení

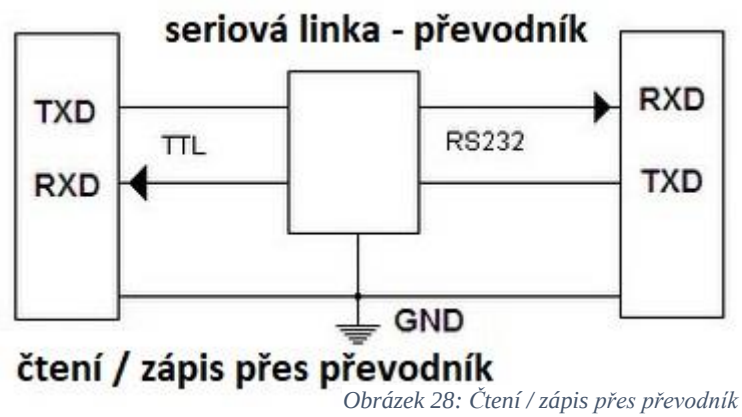
Výchozí vstup a výstup do zařízení byl zprostředkován rozhraním **UART**. Pomocí USB převodníku byla zařízení propojena přes piny **TxD**, **RxD** a GND.



Obrázek 26: UART převodník

Propojení bylo zprostředkováno programem **screen** na přenosové rychlosti **115200**.

```
screen /dev/ttyUSB1 115200
```



Obrázek 28: Čtení / zápis přes převodník

TX – odesílání zpráv

RX – přijímání zpráv

Viz příloha ***Sériová rozhraní zařízení***.

7.3 Použití frameworku Buildroot

Pro kompilaci systému pro jinou architekturu byly doinstalovány do hostitelského systému knihovny pro hostovaný systém. Do systému Debian byly doinstalovány pomocí balíčkovacího systému **dpkg.** Názvy knihoven a programů jsou rozdílně specifikovány oproti generickým označením, která jsou uvedena v teoretické části diplomové práce.

Framework byl stažen pomocí nástroje **git**.

```
git clone git://git.buildroot.net/buildroot
```

Do systému Debian Buster bylo nutné doinstalovat tyto chybějící knihovny a programy.

```
sudo apt install build-essential pkg-config gcc libncurses5-dev libncursesw5-dev
g++ rsync g++-8-multilib
```

Grafické uživatelské rozhraní konfiguračního průvodce **xconfig** vyžadovalo QT knihovny.

```
sudo apt install qt5-default qtbase5-dev
```

Rozhraní **gconfig** knihovny GTK a uživatelské rozhraní ***glade***.

```
sudo apt install libgtk2.0-dev libglib2.0-dev libglade2-dev
```

Buildroot nabízí vzory konfiguračních souborů pro jednotlivá zařízení. Podle názvu zařízení byl konfigurátor doplněn o potřebné technické údaje týkající se zařízení. Doplnění parametrů kernelu a způsob zavedení systému. Tyto hodnoty jsou neměnné a každý výrobce má vlastní pravidla, jak uvést svoji desku do provozu. [3]

Po každém bloku nejvyšší úrovně je doporučeno stav uložit pro správné vygenerování následujících závislostí. Z důvodu rozsáhlosti frameworku Buildroot jsou rozváděny pouze klíčové prvky. [3]

7.3.1 Raspberry Pi 3

Pro importování technických dat charakterizujících zařízení byl použit vzor dostupný ve frameworku Buildroot.

```
make raspberrypi3_defconfig
```

Spuštění procesu tvorby operačního systému bylo provedeno příkazem **menuconfig**.

```
make menuconfig
```

Specifikace byla již přednastavena z dostupného vzoru a nebylo nutné ji měnit.

```
Target Options
    ➢ Target Architecture (ARM – little indian)
    ➢ typ procesru (Cortex A-7)
    ➢ Floating point strategy (NEON/VFPv4)
```

V ***Nastavení sestavení*** byl zvolen název konfiguračního souboru pro uložení a byla povolena mezipaměť mezi kompilacemi pro rychlejší opětovnou kompilaci.

```
Build Options // možnosti sestavení
    ➢ path/name (dpvh_rpi3_defconfig) // pro make savedefconfig
    ➢ Enable compiler cache // pro urychlení opětovné kompilace
    ➢ gcc optimization level (optimize for size)
    ➢ libraries (shared only)
```

Obecný rozdíl mezi použitím statických a dynamických knihoven je ve velikosti. Statické knihovny jsou zahrnuty v kompilovaném programu. Program je poté velikostně větší, ale spustitelný nezávisle na přítomnosti požadovaných knihoven v systému. Takto spustitelný balík by měl pracovat efektivněji díky absenci nutnosti skládání potřebných knihoven ze systému. Při použití dynamických knihoven je výsledný program menší, ale vyžaduje požadované knihovny dostupné v systému pro jejich použití.

V základním nástroji pro stavbu Linux systému byl použit derivát knihovny C určený pro ARM architekturu. Pro podporu jazyku C++ bylo nutné povolit znaky větší než 8bitů.

```
Toolchain
    ➢ Toolchain type (Buildroot toolchain)
    ➢ C library (uClib-ng)
    ➢ Enable WCHAR support
    ➢ Enable C++ support
    ➢ Enable MMU support // Memory management unit
```

V sekci konfigurace systému byl vybrán BusyBox, který krom **Init** procesu obsahuje i většinu standardních unixových utilit.

```
System configuration
    ➢ Init system (BusyBox) // init pro ARM
    ➢ Enable root login with password
    ➢ Root password (1234)
    ➢ Root filesystem overaly directories
(board/dpvh/raspberrypi3/rootfs_overlay)
    ➢ Custom scripts to run before creating filesystem images
(board/dpvh/raspberrypi3//post-build.sh)
    ➢ Custom scripts to run after creating filesystem images
(board/dpvh/raspberrypi3/rootfs_overlay/post-image.sh)
```

Ve složce **board** byla vytvořena složka **dpvh –** *diplomová-práce-vojtech-hrdina*, obsahující konfigurační soubory a skripty pro dané zařízení. Do složky **rootfs_overlay** byl přidán konfigurační soubor pro SSH.

V konfiguraci jádra operačního systému Linux byl uveden odkaz na oficiální zdrojové soubory kernelu a identifikační soubor zařízení pro správnou konfiguraci.

```
Kernel
   ➢ Kernel version (Custom git repository)
(https://github.com/raspberrypi/linux.git)
   ➢ Custom repository version (rpi-4.19.y)
   ➢ Kernel Binary format (zImage)
   ➢ In-tree Device Tree Source file names
(broadcom/bcm2710-rpi-3-b)
   ➢ Linux kernel tools
         ○ gpio // podpora pro sběrnici
         ○ tmon // přístup k teplotním informacím zařízení
```

Ze sekce ***Firmware*** byly zahrnuty veškeré podpůrné programy a ovladače pro chod WiFi, Bluetooth a ostatních periferií.

```
Target packages
   ➢ Hardware handling
         ○ Firmware
               ▪ b43-firmware, linux firmware, rpi-firmware
               ▪ Install DTB overlays
               ▪ rpi-wifi-firmware
               ▪ eudev, minicom
```

V tomto menu je také ***Linux kernel extension***, která kromě jiných nabízí ***Real-time patch.***

Pro konfiguraci sítě v systému byly vybrány následující programy.

```
   ➢ Networking applications
         ○ connman // network manager
         ○ iproute2, iptables, iw
         ○ openssh // tool for remote login
         ○ wpa_supplicant
               ▪ Enable EAP + Enable WiFi display
               ▪ Install wpa_cli Binary
               ▪ Install wpa_client shared library
               ▪ Install wpa_passphrase Binary
```

Velikost souborového systému byla nastavena podle velikosti vytvořeného operačního systému.

```
File systém images
    ➢ ext filesysytem
    ➢ exact sizes (200MB)
```

Vytvořený konfigurační soubor je dostupný v adresáři **config**. Nahrání do pracovního konfiguračního souboru **.config** je provedeno následující příkazem.

```
make dpvh_rpi3_defconfig
```

Proces generování je spuštěn příkazem make.

```
make
```

7.3.1.1 Nahrání systému na paměťové medium

Po připojení paměťové microSD karty k PC se provede automatické připojení pro prohlížení obsahu karty. Nejprve je nutné kartu odpojit a následně obraz na medium nakopírovat. Vytvořený obraz disku **sdcard.img** se nachází v adresáři **`buildroot/output/image`**.

```
sudo umount /dev/sdX{1,2}
sudo dd bs=4M if=sdcard.img of=/dev/sdX conv=sync
```

7.3.1.2 Konfigurace WiFi adaptéru

V spuštěném systému se provádí připojení do sítě pomocí instalovaného klienta ***ConnMan***.

```
connmanctl enable wifi // povolení wifi adaptéru
connmanctl // spuštění průvodce
    ➢ scan wifi // zjištění dostupných sítí
    ➢ services // zobrazení nalezených sítí
    ➢ agent on // nástroj k odchytávání uživatelských požadavků
    ➢ connect wifi_* // připojení k síti
    ➢ quit // ukončení průvodce
```

7.3.2 BeagleBone Black

Pro zařízení z dílen Texas Instrument je v nástroji Buildroot dostupný konfigurační soubor pro základní charakteristiku zařízení pro správné nastavení systému.

Pro použití vzoru byl použit příkaz ***make *_defconfig***.

```
make beaglebone_defconfig
```

Konfigurace systému byla spuštěna příkazem **menuconfig**.

```
make menuconfig
```

Specifikace cílového zařízení je převzata ze vzoru dostupného frameworkem.

```
Target options
    ➢ Target Architecture Variant (cortex-A8)
    ➢ Floating point stratégy (VFPv3-D16)
```

V nastavení kompilace byla povolena mezipaměť pro rychlejší iteraci při vývoji.

```
Build options
    ➢ Enable compiler cache
($HOME/.buildroot.ccache)
```

Z klíčových vlastností v ***Toolchain*** sekci byla zvolena redukovaná verze knihovny C pro ARM systémy a WCHAR, které umožňují použití znaků složených z více než 8bitů.

```
Toolchain
    ➢ C library (uClib-ng)
    ➢ Enable WCHAR support
```

Přístup do systému byl nastaven v sekci ***Konfigurace systému***.

```
System configuration
    ➢ Enable root login with password
    ➢ Run a getty after boot
```

V sekci ***Jádro systému*** byl nakonfigurován aktuálně podporovaný kernel od výrobce desky a označení desky pro jeho správnou konfiguraci.

```
Kernel
    ➢ Kernel Version (4.19.79-ti-r30)
    ➢ Build a Device Tree Blob (amx335x-boneblack)
```

V sekci ***Balíčky pro cílové zařízení*** byly vybrány ovladače pro hardware a programy pro síťovou komunikaci.

```
Target packages
    ➢ Hardware handling
        o linux-firmware
        o setserial
    ➢ Networking applications
        o iproute2
        o iptables
```

Systém byl posazen na standardní linuxový souborový systém a velikost stanovena na 512MB.

```
Filesystem images
    ➢ root filesystem (ext4)
    ➢ exact size (512MB)
```

Pro zavedení operačního systému byl použit **U-Boot**.

```
Bootloaders
    ➢ U-Boot (04/2020)
```

Vytvořený konfigurační soubor pro vygenerování operačního systému je dostupný ve složce **configs**. Načtení je provedeno pomocí příkazu make **dpvh_beaglebone_defconfig**.

```
make dpvh_beaglebone_defconfig
```

Spuštění kompilace je provedeno příkazem **make**.

```
make
```

Nahrání systému na medium je popsáno v kapitole 7.3.1.1 **Nahrání systému na paměťové medium**.

7.3.3 Orange Pi Zero

Pro nahrání vzoru se základními parametry charakterizující zařízení, ze složky ***configs,*** byl použit příkaz ***make*** a jako parametr název zařízení.

```
make orangepi_zero_defconfig // konfigurační soubor (z /configs)
```

Konfigurační průvodce byl spuštěn příkazem **menuconfig**.

```
make menuconconfig
```

Specifikace cílového zařízení byla ponechána.

```
Target Options // základní nastavení zařízení
    ➢ Target Architecture (ARM – little indian)
    ➢ typ procesru (Cortex A-7)
    ➢ Floating point strategy (VFPv4)
```

V nastavení buildu byl pojmenován konfigurační soubor na **dpvh_orz_defconfig** a povolena mezipaměť mezi kompilacemi pro rychlejší opětovné kompilace.

```
Build Options // možnosti kompilace
    ➢ path/name (dpvh_orz_defconfig)
    ➢ Enable compiler cache
    ➢ gcc optimization level (optimize for size)
    ➢ libraries (shared only)
```

V sadě programovacích nástrojů **toolchain** byl použit nativní ***Buildroot toolchain***. Knihovna jazyka C byla zvolena odvozená verze pro ARM procesory.

```
Toolchain
    ➢ Toolchain type (Buildroot toolchain) // nativní toochain
    ➢ C library (uClib-ng) // knihovna jazyka C pro ARM
    ➢ Enable WCHAR support // wide char
    ➢ Enable C++ support
    ➢ Enable MMU support // k převodu virtuální adresy na fyzickou
```

V konfiguraci systému byl zvolen typ **init** procesu implementovaný ze sady procesů **BusyBox**, který má v sobě vestavěné implementace standardních unixových příkazů. Bylo povoleno přihlášení za **roota** a přítomnost terminálu na sériovém rozhraní.

```
System configuration
    ➢ Init system (BusyBox) // init pro ARM
    ➢ Enable root login with password
    ➢ Root password () // přihlášení bez hesla
```

V sekci **Kernel**, jádro operačního systému Linux, je definice verze jádra, která bude stažena a zkompilována. Konfigurační soubor je dodávaný frameworkem.

```
Kernel
   ➢ Kernel version (5.3.8)
   ➢ Additional configuration fragment files
(board/orangepi/orangepi-zero/linux-extras.config)
```

Základní sada balíčků pro zařízení byla rozšířena o programy zefektivňují práci se systémem.

```
Target Package
   ➢ Firmware
         o armbian-firmware
         o XR819 Wifi
   ➢ Networking applications
         o connman // network manager
         o iproute2
         o wpa_suppliciant
   ➢ xr819-xradio
```

Velikost souborového systému je nastavena na velikost operačního systému s rezervou pro odkládání dat procesy.

```
Filesystem images
   ➢ exact size (60MB)
```

Pro zavedení sestaveného systému byl použit zavaděč **U-boot.**

```
Bootloaders
   ➢ U-Boot
   ➢ version (2019.10)
```

Vytvořený konfigurační soubor je uložen v adresáři **config**. Nahrání pro kompilaci je prováděno příkazem **make dpvh_orz_defconfig**.

```
make dpvh_orz_defconfig
```

Proces kompilace je spuštěn příkazem **make**.

Nahrání systému na medium je popsáno v kapitole 7.3.1.1 **Nahrání systému na paměťové medium** a konfigurace WiFi adaptéru v kapitole 7.3.1.2 **Konfigurace WiFi adaptéru**.

7.4 Použití frameworku OpenADK

Framework byl stažen z oficiálních stránek pomocí nástroje **git**.

```
git clone git://openadk.org/git/openadk
```

Do systému Debian Buster museli být doinstalovány chybějící programy.

```
sudo apt install binutils gcc libncurses* libncurses5-dev zlib1g gzip perl tar
wget swig ncurses
```

OpenADK byl instalován druhý v pořadí a velké množství požadovaných programů již bylo v systému obsaženo.

Operační systémy vytvořené ve frameworku OpenADK jsou ve výchozím použití nastavena pouze pro čtení. Operace vytvoření, nebo úprava souboru, není v root souborovém systému umožněna. Pokus skonči hláškou a operace se neprovede.

7.4.1 Raspberry PI 3

Proces stavby operačního systému byl zahájen příkazem **menuconfig**.

```
make menuconfig
```

V první části byl vybrán operační systém, architektura a knihovna jazyka C.

```
Operating Systém (Linux)
Hardware Configuration
    ➢ Architectura (arm)
    ➢ Systém (Raspberry PI 3)
    ➢ CPU (cortex-a53)
C Library configuration
    ➢ Target C library (uClib-ng)
```

Na vyžádání konfiguračního průvodce frameworku byl konfigurátor při každé změně z výše uvedené nabídky uložen.

Veškeré kompilace programů budou prováděny z dané knihovny C. Při změně je nutný celý proces spustit od začátku. Výchozí shell pro systém byl zvolen **mksh**.

```
Package selection
    ➢ Base systém
        o Boot Loaders (bcm28xx-bootloader)
        o Base Libraries (libgcc)
```

Firmware pro WIFI adaptér byl implementován **b43-firmaware**.

```
    ➢ Systém
        o Firmware for Hardware Devices (b43-firmware)
```

Nástroj pro nastavení WIFI adaptéru byl zvolen **iw** a **wpa_supplicant**.

```
    ➢ Networking
        o Wireless (iw, wpa_supplicant)
```

V nastavení běhu systému je povoleno spuštění terminálu v seriál i VGA konzoli.

```
    ➢ Runtime configuration
        o start getty on VGA/seriál console
```

Verze kernelu a způsob stažení jsou nastaveny v sekci Linux Kernel.

```
Linux kernel configuration
    ➢ github.com/raspberrypi/linux.git
    ➢ version (a209214180775)
```

Pro bezdrátovou komunikaci byl povolen *Broadcom wireless* driver,

```
➢ Advanced Linux Kernel configuration
    o Wireless network card support (Broadcom brcmfac driver)
```

modul pro podporu klávesnice a

```
    o Input devices (USB Human Interactive Device support)
```

povolení pro síťovou komunikaci TCP/IP.

```
    o Enable TCP/IP support
```

V nastavení Toolchain byl zvolen kompilační nástroj, Binutils a verze kompilátoru GCC.

```
Toolchain settings
➢ Binutils version (2.35)
➢ Compiler (gcc)
➢ GCC version (7.5.0)
```

V sekci globální konfigurace byla povolena **cache** a více vláknové zpracování.

```
Global settings
➢ Use ccache to speedup recompilation
➢ Enable paralel building
```

Připravený konfigurační soubor je dostupný v adresáři **OpenADK/configs/RP3**. Před nakopírováním souboru a spuštěním kompilačního procesu je nutné vyvolat konfiguračního průvodce příkazem **make menuconfig** a poté uložit.

Kompilace systému je spuštěna příkazem **make**.

```
make
```

7.4.1.1 Nahrání systému na paměťové medium

Nahrání vytvořeného obrazu na medium je prováděno pomocí předpřipraveného skriptu, který rozšíří souborový systém na celé paměťové medium.

Nahrání je nutné provádět s administrátorskými právy a s odpojeným paměťovým zařízením.

```
sudo umount /dev/sdX1 /dev/sdX2
```

Pro úspěšné kompilaci systému se zobrazí skript pro nakopírování systému na SD kartu.

```
sudo ./scripts/install.sh raspberry-pi3 /dev/sdX
/home/vojta/diplomka/openadk_rp/firmware/raspberry-pi3_uclibc-
ng_cortex_a53_hard_eabihf_arm/raspberry-pi3-uclibc-ng-archive+kernel.tar.xz
```

7.4.2 BeagleBone Black

Konfigurační průvodce byl spuštěn příkazem **menuconfig**.

```
make menuconfig
```

Následně byl vybrán operační systém Linux, architektura a cílový systém.

```
Operation Systém (Linux)
Hardware configuration (arm, BeableBone, cortex-a8)
```

Knihovna C byla vybrána **uClibc-ng**, odvozená od knihovny *libc*.

```
C Library configuration
    ➢ Target C library (uClibc-ng)
```

V sekci výběru balíčků byl zvolen zavaděč a **GCC** knihovna.

```
Package selection
    ➢ Base system
        ○ Boot loaders (u-boot)
        ○ Base Libraries (libgcc)
```

Přístup do systému byl nastaven v sekci ***Správce systému***.

```
Runtime configuration
    ➢ enable login as root
    ➢ root password (abcdef)
    ➢ size of /tmp in memory (1024kb)
```

Byl povolen konzolový přístup přes terminál i na VGA výstup. Síťová konfigurace byla nastavena na přidělování IP adresy pomocí DHCP.

```
    ➢ start getty on VGA/serial console
    ➢ seriál console speed (115200)
    ➢ eth0 Configuration (DHCP)
```

Verze Kernelu a podpora USB zařízení byla nastavena v ***Rozšířeném nastavení kernelu***.

```
Linux Kernel configuration
    ➢ Kernel version (4.19.99)
    ➢ Input devices (USB Human Interactive Device support)
    ➢ Network Support (Enable TCP/IP)
```

Připravený konfigurační soubor je dostupný v adresáři **OpenADK/configs/BBB**. Před nakopírováním a spuštěním kompilačního procesu je nutné vyvolat konfiguračního průvodce příkazem **make menuconfig** a poté uložit. Proces generování je spuštěn příkazem **make**.

Framework OpenADK nenabízí nahrání systému na paměťové medium pro zařízení BeagleBone Black.

7.4.3 Orange Pi

Konfigurační průvodce návrhu operačního systému Linux pro zařízení Orange Pi Zero bylo vyvoláno příkazem **menuconfig**.

```
make menuconfig
```

Zvolením architektury, určením zařízení a typu procesoru byl nastaven základ pro tvorbu systému.

```
Hardware configuration
    ➢ Architectura (arm)
    ➢ System (Orange Pi Zero)
    ➢ CPU (Cortex A-7)
```

Knihovna jazyka C byla použita odlehčená verze pro ARM architekturu. Krom strategie plovoucí čárky byla stanovena pravidla pro použití knihoven.

```
C Library configuration
    ➢ Target C Library (uClib-ng, 1.0.34)
    ➢ Float configuration (hard-float)
    ➢ Library support (only shared library and link dynamically)
```

Zavaděč byl zvolen **U-boot** a výchozí shell **mksh**.

```
Package selection
    ➢ Base system
        o Boot Loaders (u-boot)
        o Shells (mksh)
```

V sekci ***Správce běhu zařízení*** byl povolen terminál na konzoli a byla nastavena standardní rychlost přenosu.

```
Runtime configuration
    ➢ start getty on seriál console
    ➢ seriál console speed (115200)
```

Připojení do sítě je nastaveno na automatickou konfigurací pomocí DHCP.

```
    ➢ eth0 Configuration (DHCP)
```

V ***Konfiguraci kernelu*** v podsekci ***Rozšířené nastavení*** byla implementována podpora pro ethernet a bezdrátovou síť.

```
Linux Kernel COnfiguration
   ➢ Advanced Linux Kernel configuration
         o Network device support
               ▪ Ethernet network device support (DE2104X)
               ▪ Wireless network card support
```

Přidána byla podpora pro USB zařízení a byl povolen TCP/IP protokol.

```
         o USB support (USB storage, EHCI controllers)
         o Seriál device support (8250 seriál driver)
         o Input device (USB Human Interactive Device support)
         o Network support (TCP/IP)
```

Výchozí kompilátor pro proces byl zvolen **gcc**.

```
Toolchain settings
   ➢ Compiler (GCC)
   ➢ GCC version (8.3.0)
```

Při vývoji byla použita mezipaměť mezi kompilacemi a povolené paralelní zpracování úlohy.

```
Global settings
   ➢ Use ccache to speedup recompilation
   ➢ Enable parallel building
```

Připravený konfigurační soubor je dostupný v adresáři **OpenADK/configs/ORZ**. Před nakopírováním a spuštěním kompilačního procesu je nutné vyvolat konfiguračního průvodce příkazem **make menuconfig** a poté uložit. Proces generování systému je spuštěn příkazem **make**. Nahrání obrazu systému na paměťové medium není funkční. Bez rozsáhlých dodatečných úprav zařízení nenabootuje.

8 Porovnání vytvořených systému

U vytvořených operačních sytému byly provedeny testy na funkcionalitu a výkonnostní testy na efektivitu systému. Při porovnání s univerzálními dodávanými systémy byly vytvořeny deriváty, které se lišily velikostí a způsobem práce s nimi.

Hlavním výsledkem bylo docílení systému, který je oproštěn od složitosti generických robustních řešení a tím zefektivnění základních rysů embedded systémů jako je jednoduchost, efektivita a velikost samotného systému.

Systémy vytvářené ve frameworku Buildroot disponovaly plnohodnotným zázemím pro vytvoření široko spektrálního systému připomínajícího generické dodávané řešení postavené na robustní základně systému Debian. Cílem bylo udržet systém co nejmenší a nejjednodušší s implementací pouze prvků nezbytně nutných pro dané zařízení.

Řešení zprostředkované frameworkem OpenADK se stavilo k striktně minimalistické politice vytvoření sytému umístěného na nepřepisovatelný souborový systém, který tak pasivně zajišťoval vysokou bezpečnost a bezporuchovost běhu zařízení. Pro zařízení Orange Pi Zero a BeagleBone Black byl framework zastaralý a nenabízel podporu pro implementaci operačního systému na paměťové medium.

8.1 Raspberry Pi 3

Oficiálně dodávaný systém Raspbian lite zabírá téměř 1.4 GB a běh systemu zabere 70 MB operační paměti. Na pozadí běží 111 procesů.

Systém vytvořený ve frameworku Buildroot zabírá 133 MB a po zapnutí zařízení si alokuje 56 MB operační paměti. Spuštěných procesů je osmdesát osm.

Ve frameworku OpenADK byl vytvořen systém nejmenší s nejmenší softwarovou vybaveností. Velikost souborového systému pro operační systém je 65 MB a využívá 13.5 MB operační paměti. Po spuštění zařízení v systému běží 64 procesů.

Raspberry Pi 3	Raspbian	Buildroot	OpenADK
velikost systému	1388 MB	132.8 MB	64.9 MB
použitá paměť systému	70,3 MB	55,5 MB	13,5
vytížení CPU	~1 %	~1 %	~0 %
procesů v systému	111	88	64
benchmark – WH komprimace	-	+24%	+32%
benchmark – WH dekomprimace	-	+13%	+10%
benchmark – Int – writing	-	+13%	+33%
benchmark – Int – reading	-	+10%	+23%
benchmark – Int – memory	-	+12%	+25%
benchmark – Float – writing	-	+14%	+31%
benchmark – Float – reading	-	+6%	+25%
benchmark – Float – memory	-	+6%	+14%

Systém vygenerovaný ve frameworku Buildroot vykazuje na základě měření nárůst výkonu v průměru o **12%** a systém vygenerovaný ve frameworku OpenADK vykazuje systém efektivnější o **24%**.

8.2 BeagleBone Black

Dodávaný systém pro zařízení z dílen BeagleBone postavený na systému Debian zabírá 2,5 GB na paměťovém mediu. Po nabootování systému je velikost alokované paměti 55 MB. Spuštěných procesů v systému po zapnutí zařízení je 89.

Vygenerovaný systém frameworkem zabírá 37 MB a alokuje si 11 MB operační paměti. Procesů systému v nečinném stavu je 49.

BeagleBone Black	Beagle Debian	Buildroot
velikost systému	2.5 GB	37.2 MB
použitá paměť systému	54.6 MB	10.6 MB
vytížení CPU	~2%	~0%
procesů v systému	89	49
benchmark – WH komprimace	-	+16%
benchmark – WH dekomprimace	-	+14%
benchmark – Int – writing	-	+1%
benchmark – Int – reading	-	+2%
benchmark – Int – memory	-	+2%
benchmark – Float – writing	-	0
benchmark – Float – reading	-	+2%
benchmark – Float – memory	-	+9%

Systém navržený ve frameworku Buildroot vykazuje podle výsledků prováděných testů nárust efektivity o **5%**. Framework OpenADK neposkytuje funkční řešení generování systému pro zařízení BeagleBone. Rozsáhlé úpravy se neslučují s použitím základních prvků systému vygenerovaných frameworkem.

8.3 Orange Pi Zero

Velikost operačního systému Debian, upraven výrobcem zařízení Orange Pí, zabírá 872 MB na paměťovém mediu. Po spuštění systému je alokováno 51 MB operační paměti a počet procesů v nečinném systému po zapnutí zařízení je 102.

Systém vytvořený frameworkem Buildroot je velký 172 MB a zabírá tedy jen 20% velikosti oficiálně dodávaného systému. Po spuštění systému je zabráno 19 MB operační paměti. Počet běžících procesů v systému je 73.

Orange PI Zero	Debian OrangePi	Buildroot
velikost systému	872 MB	171.5 MB
použitá paměť systému	51 MB	19 MB
vytížení CPU	~1 %	~2 %
procesů v systému	102	73
benchmark – komprese	-	+14%
benchmark – dekomprese	-	+9%
benchmark – INT	-	+27%
benchmark – FLOAT	-	+17%
benchmark – Int – memory	-	+42%
benchmark – Float – writing	-	+41%
benchmark – Float – reading	-	+16%
benchmark – Float – memory	-	+15%

Provedené testy vykazují **22%** nárustu výkonu vygenerovaného systému oproti oficiálně dodávanému. Framework OpenADK neposkytuje funkční řešení pro nahrání systému na paměťové medium a rozsáhlé úpravy se neslučují se zachováním hlavních pilířů z vygenerovaného systému.

9 ZÁVĚR

Výsledkem diplomové práce jsou vygenerované operační systémy ve frameworcích Buildroot a OpenADK pro zařízení Raspberry Pi 3, BeagleBone Black a Orange Pi ZERO.

V úvodu jsou pospána jednotlivá zařízení a jejich možný způsob použití. Je představena koncepce linuxových systémů na architektuře ARM a porovnána s platformou určenou pro desktopová zařízení.

Před zahájením procesu generování linuxových systémů bylo popsáno použití jednotlivých frameworků a zmíněny typy pro jejich efektivní využití.

Pomocí nástrojů pro generování linuxových systémů byly navrženy alternativy k běžně dodávaným systémům od výrobců daných zařízení.

Generování operačního systému začínalo od specifikace základní desky, konfigurace systémového jádra, implementace ovladačů pro dané zařízení, až po systémové a uživatelské programy pro řízení zařízení. Vytvořené systémy byly nakonfigurovány pro základní použití a otestovány na základní funkcionalitu.

Vygenerované operační systémy jsou velikostně menší, zabírají méně operační paměti a podle výsledku testů pracují efektivněji.

10 PŘÍLOHY

10.1 Benchmark – Raspberry Pi 3

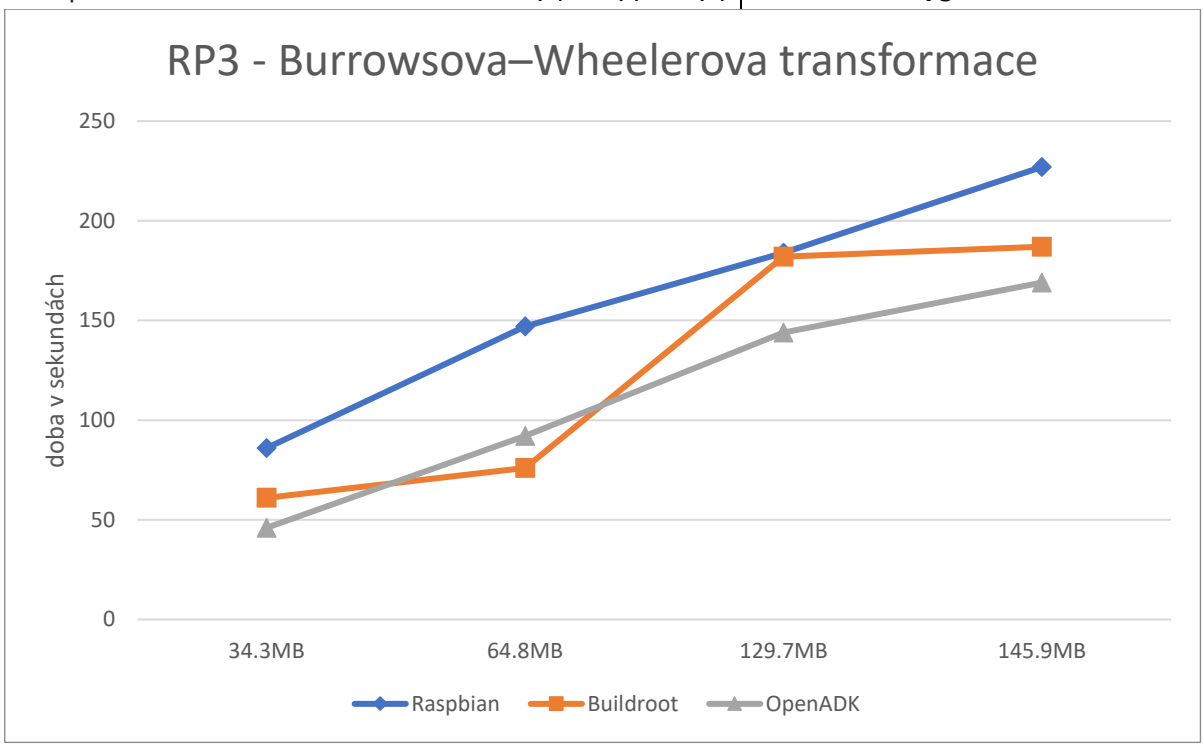
Burrowsova–Wheelerova transformace – komprimace / dekomprimace

soubor vytvořen pomocí nástroje **dd** o velikosti *count*=**počet** *bs*=**náhodných_bajtů**

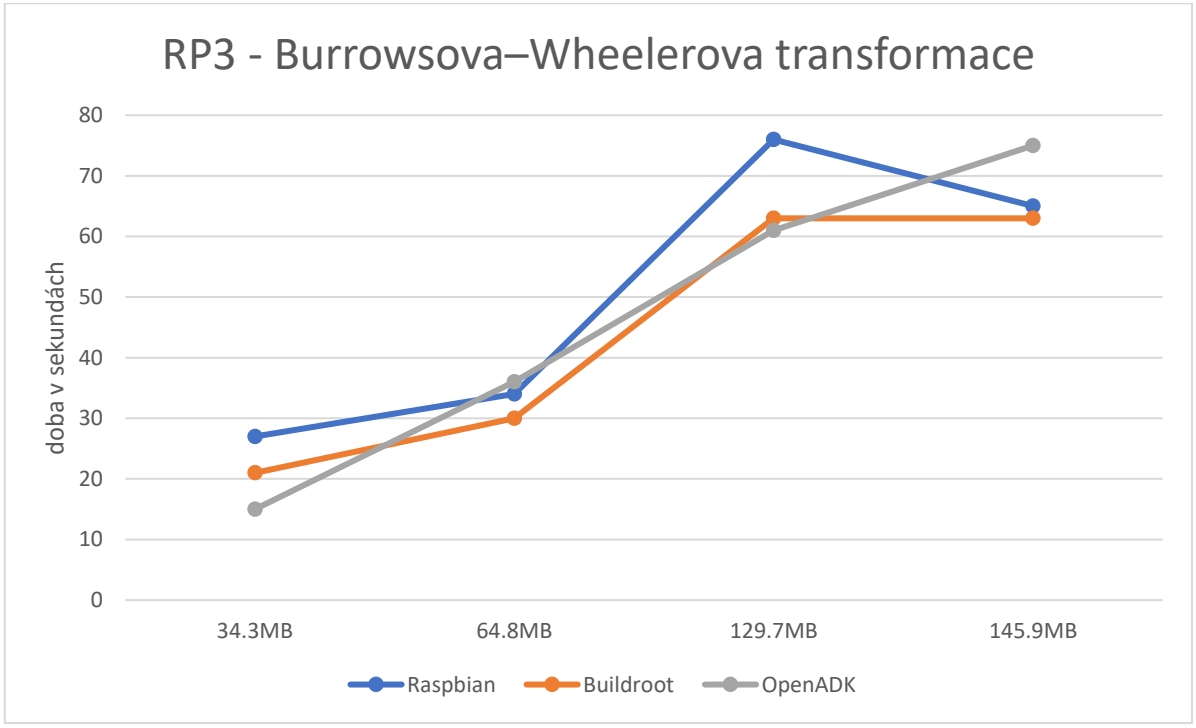
```
dd if=/dev/random of=myrandom bs=x count=y
```

	měřené hodnoty			~
`bs=8000 count=4500 // 34.3 MB`				
➢ komprimace				
○ Raspbian	87	85	86	**86**
○ Buildroot	62	58	63	**61**
○ OpenADK	46	45	47	**46**
➢ dekomprimace				
○ Raspbian	28	26	27	**27**
○ Buildroot	21	19	23	**21**
○ OpenADK	15	14	16	**15**
`bs=8000 count=8500 // 64.8 MB`				
➢ komprimace				
○ Raspbian	148	146	147	**147**
○ Buildroot	75	75	78	**76**
○ OpenADK	92	91	93	**92**
➢ dekomprimace				
○ Raspbian	33	35	34	**34**
○ Buildroot	29	27	34	**30**
○ OpenADK	34	36	38	**36**
`bs=8000 count=17000 // 129.7 MB`				
➢ komprimace				
○ Raspbian	179	185	188	**184**
○ Buildroot	182	185	179	**182**
○ OpenADK	132	148	152	**144**
➢ dekomprimace				
○ Raspbian	78	75	75	**76**
○ Buildroot	64	60	65	**63**
○ OpenADK	60	63	60	**61**
`bs=18000 count=8500 // 145.9 MB`				
➢ komprimace				
○ Raspbian	229	230	222	**227**
○ Buildroot	188	185	188	**187**
○ OpenADK	168	173	166	**169**
➢ dekomprimace				
○ Raspbian	62	65	68	**65**
○ Buildroot	60	65	64	**63**

○ OpenADK	74	77	74	**75**

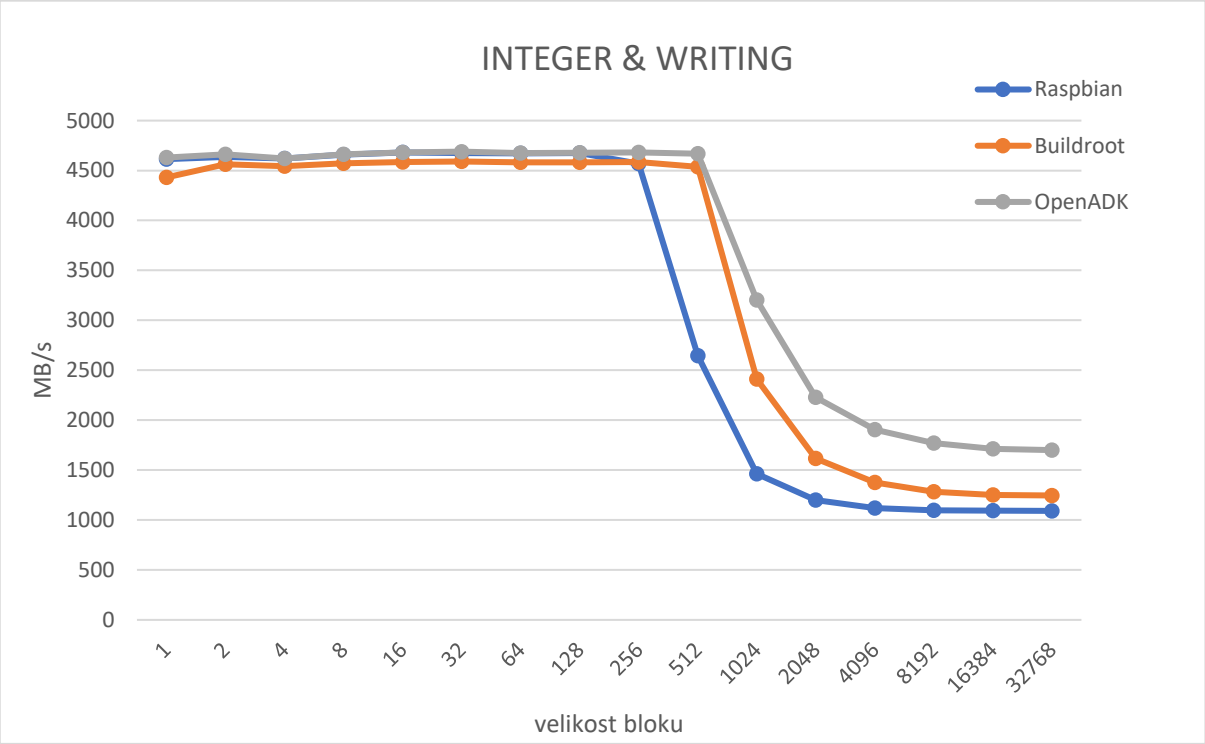


Obrázek 29: Benchmark Raspberry Pi 3 - BW (1)



Obrázek 30: Benchmark Raspberry Pi 3 - BW (2)

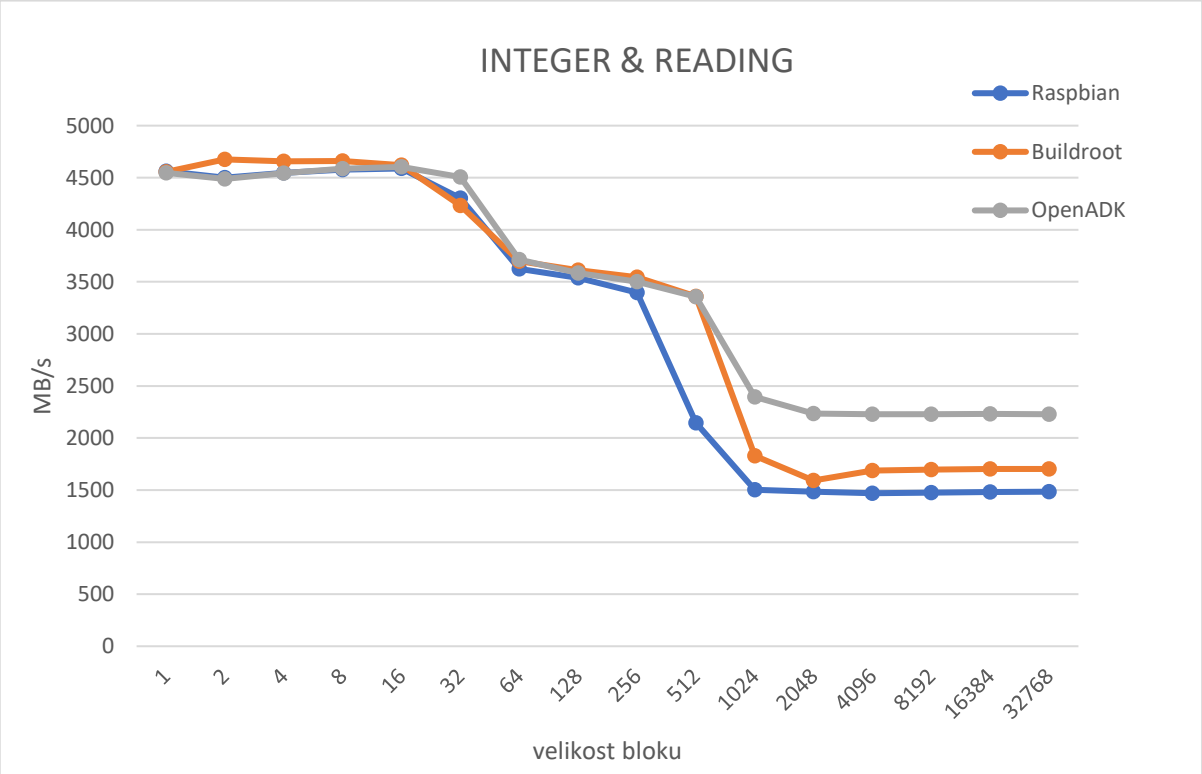
Raspberry Pi 3 - RAMSPEED - INTEGER & WRITING MB/s			
velikost bloku (kb)	Raspbian	Buildroot	OpenADK
1	4612.38	4431.35	4631.42
2	4637.58	4561.82	4663.25
4	4620.2	4543.5	4618.95
8	4659.93	4572.7	4661.68
16	4680.6	4585.98	4682.8
32	4674.27	4590.46	4689.18
64	4671.82	4582.11	4675.62
128	4675.48	4582.88	4679.47
256	4569.22	4585.05	4680.57
512	2644.56	4537.95	4669.27
1024	1463.36	2412.63	3202.08
2048	1199.4	1617.93	2228.69
4096	1121.03	1375.95	1903.42
8192	1098.56	1281.97	1769.02
16384	1094.25	1250.12	1712.51
32768	1091.46	1245.86	1700.44



Obrázek 31: Benchamrk Raspberry Pi 3 - Integer writing

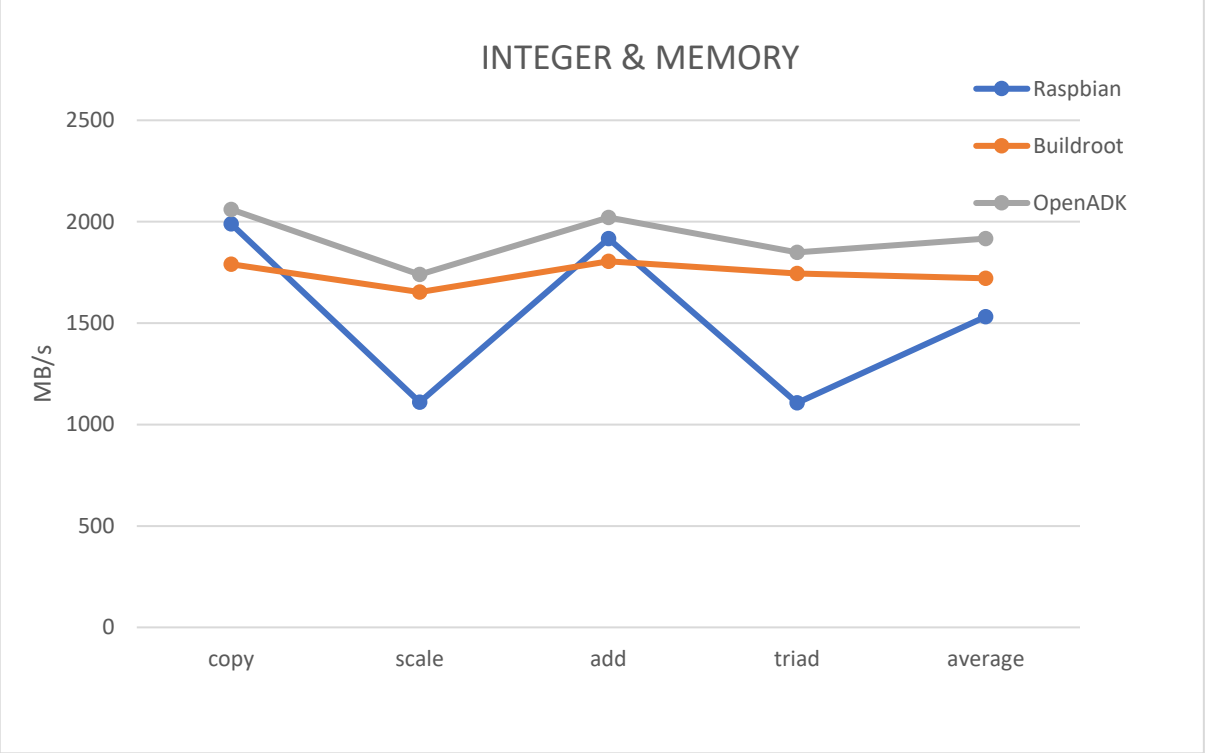
Raspberry Pi 3 - RAMSPEED - INTEGER & READING MB/s

velikost bloku (kb)	Raspbian	Buildroot	OpenADK
1	4560.63	4555.18	4545.15
2	4498.82	4675.49	4487.92
4	4547.43	4657.21	4541.82
8	4576.96	4661.28	4587.24
16	4588.47	4618.76	4603.44
32	4304.35	4233.33	4506.52
64	3625.21	3699.38	3713.62
128	3536.88	3613.15	3582.86
256	3396.5	3544.89	3500.89
512	2145.07	3358.74	3357.82
1024	1503.7	1829.8	2393.11
2048	1485.61	1591.78	2233.13
4096	1469.7	1685.68	2228.52
8192	1474.9	1696.86	2228.45
16384	1482.7	1702.46	2229.99
32768	1482.93	1701.33	2229.39



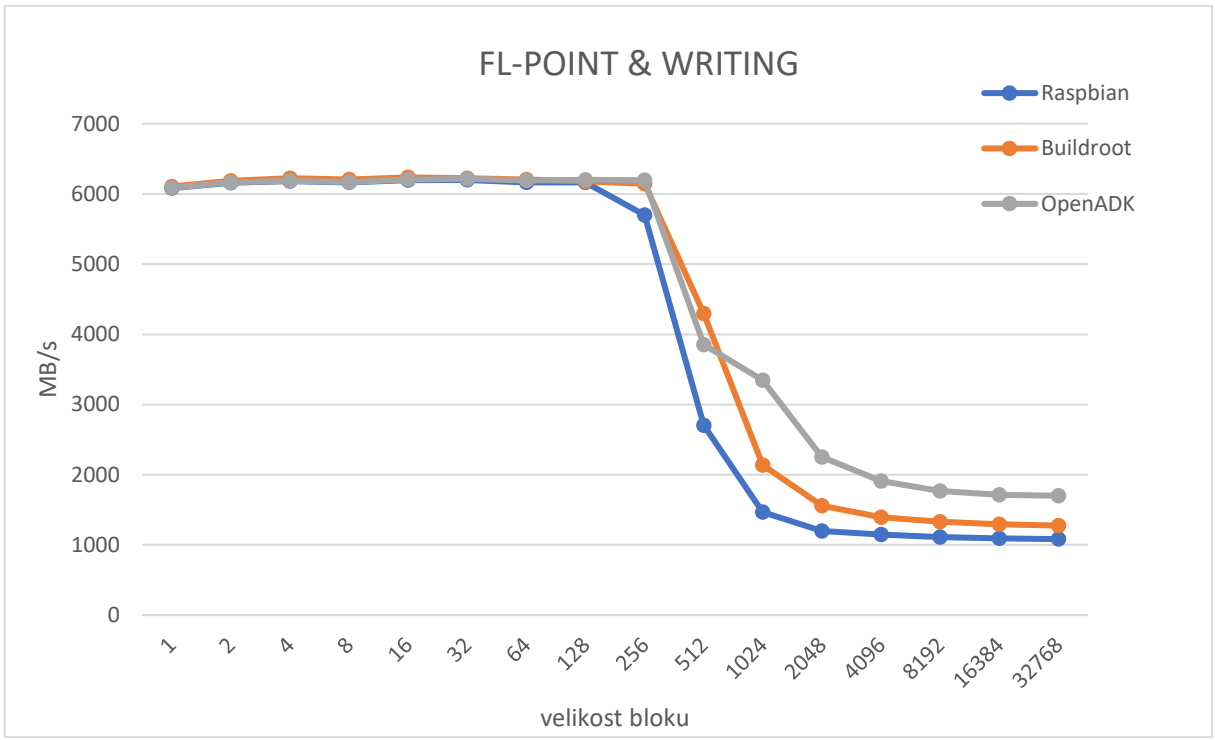
Obrázek 32: Benchmark Raspberry iI 3 – Integer reading

Raspberry Pi 3 - RAMSPEED - INTEGER & MEMORY MB/s			
	Raspbian	Buildroot	OpenADK
copy	1989.43	1789.7	2060.73
scale	1110.88	1652.96	1739.42
add	1916.59	1804.75	2021.44
triad	1106.54	1744.39	1848.12
average	1530.86	1720.41	1917.43



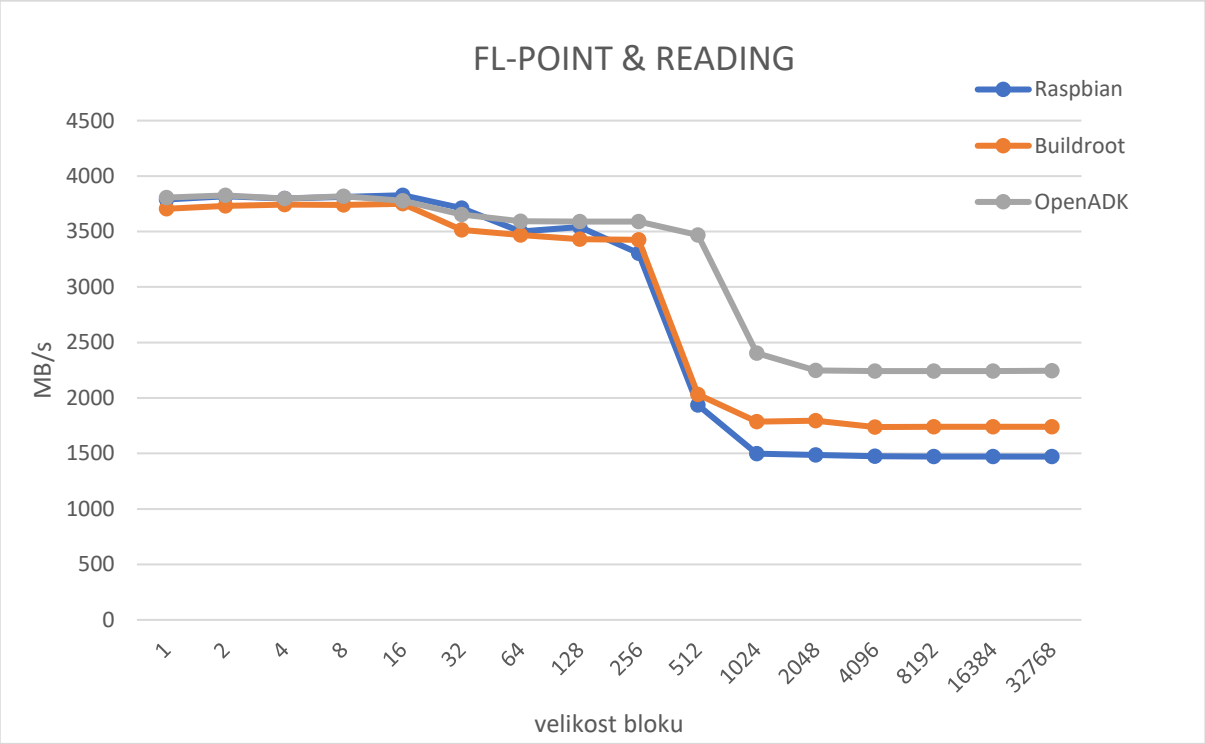
Obrázek 33: Benchmark Raspberry Pi 3 - Integer memory

Raspberry Pi 3 - RAMSPEED – FL-POINT & WRITING MB/s			
velikost bloku (kb)	Raspbian	Buildroot	OpenADK
1	6080.87	6105.97	6086.05
2	6158.05	6184.88	6163.02
4	6180.18	6223.26	6187.08
8	6164.75	6206.56	6169.99
16	6194.9	6234.81	6204.43
32	6198.15	6222.66	6214.97
64	6161.72	6203.14	6189.25
128	6163.2	6177.14	6200.87
256	5697.7	6143.52	6197.27
512	2701.4	4293.96	3849.62
1024	1469.11	2139.52	3345.35
2048	1198.3	1557.3	2251.49
4096	1147.07	1393.62	1908.98
8192	1110.69	1329.75	1769.41
16384	1092.67	1294.87	1712.31
32768	1082.34	1274.64	1700.29



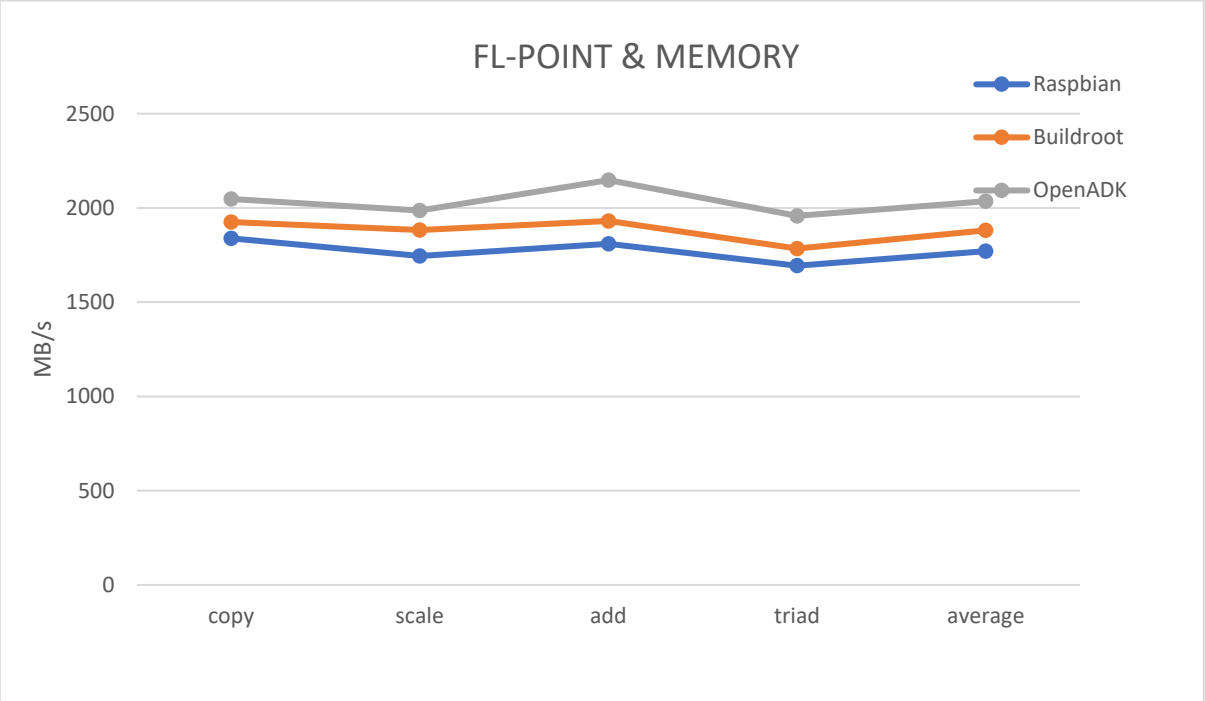
Obrázek 34: Benchmark Raspberry Pi 3 - Fl-point writing

Raspberry Pi 3 - RAMSPEED – FL-POINT & READING MB/s			
velikost bloku (kb)	Raspbian	Buildroot	OpenADK
1	3788.87	3704.74	3806.17
2	3817.61	3732.14	3825.6
4	3797.09	3742.64	3798.88
8	3812.56	3738.75	3816.85
16	3827.41	3750.09	3776.6
32	3710.32	3514.73	3653.54
64	3499.93	3469.91	3593.9
128	3542.14	3430.98	3590.79
256	3304.9	3424.09	3588.72
512	1935.97	2032.85	3467.81
1024	1499.16	1786.82	2404.37
2048	1486.53	1795.59	2247.71
4096	1473.99	1738.17	2242.44
8192	1472.44	1739.35	2242.75
16384	1471.85	1740.14	2242.73
32768	1471.66	1741.1	2244.26



Obrázek 35: Benchmark Raspberry Pi 3 - Fl-point reading

Raspberry Pi 3 - RAMSPEED – FL-POINT & MEMORY MB/s			
velikost bloku (kb)	Raspbian	Buildroot	OpenADK
copy	1838.76	1925.18	2047.95
scale	1744.4	1883.42	1986.04
add	1808.83	1930.95	2147.47
triad	1693.5	1784.42	1957.99
average	1771.373	1880.993	2034.86



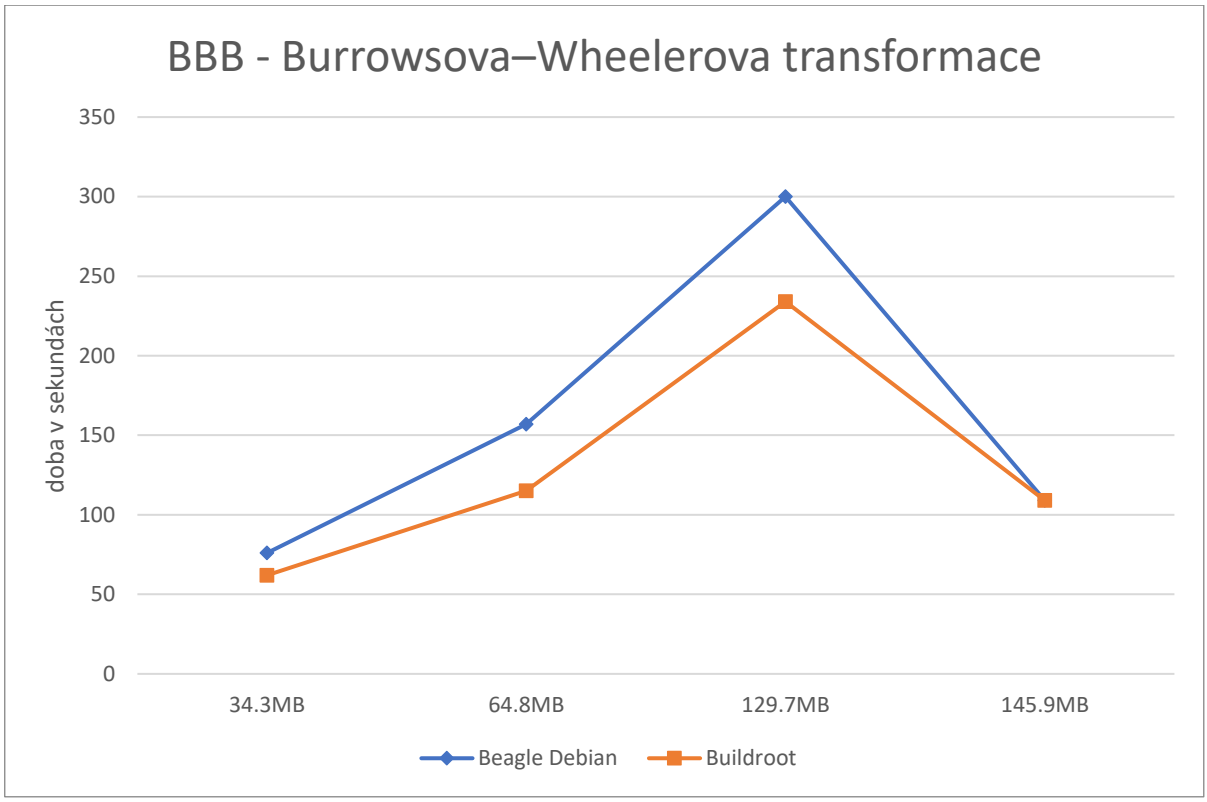
Obrázek 36: Benchmark Raspberry Pi 3 - Fl-point memory

10.2 Benchmark – BeagleBone Black

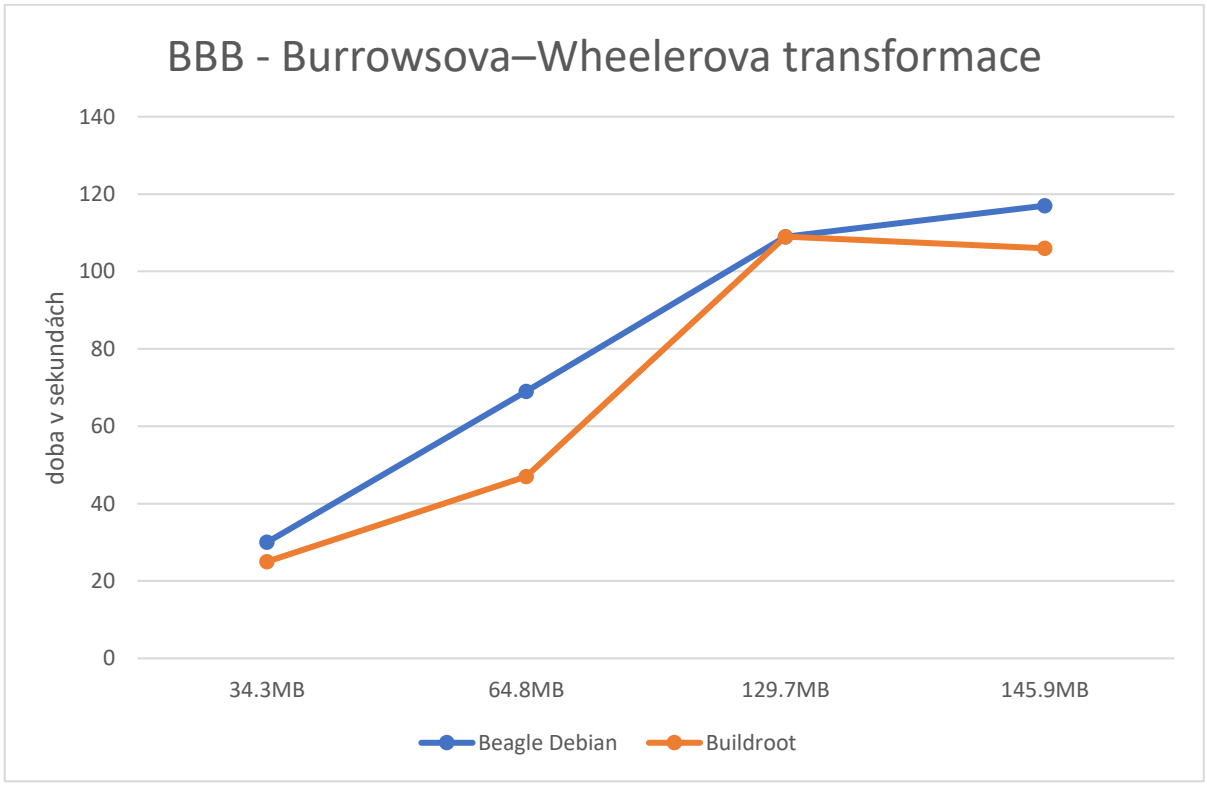
Burrowsova–Wheelerova transformace – komprimace / dekomprimace

soubor vytvořen pomocí nástroje **dd** o velikosti *count*=**počet** *bs*=**náhodných_bajtů**
`dd if=/dev/random of=myrandom bs=x count=y`

	měřené hodnoty			~
`bs=8000 count=4500 // 34.3 MB`				
➢ komprimace				
○ Beagle Debian	77	77	74	**76**
○ Buildroot	61	62	63	**62**
➢ dekomprimace				
○ OrangePi Debian	29	32	29	**30**
○ Buildroot	25	26	24	**25**
`bs=8000 count=8500 // 64.8 MB`				
➢ komprimace				
○ Beagle Debian	158	155	158	**157**
○ Buildroot	116	117	112	**115**
➢ dekomprimace				
○ OrangePi Debian	69	70	68	**69**
○ Buildroot	47	46	48	**47**
`bs=8000 count=17000 // 129.7 MB`				
➢ komprimace				
○ Beagle Debian	297	299	304	**300**
○ Buildroot	233	235	234	**234**
➢ dekomprimace				
○ OrangePi Debian	108	111	108	**109**
○ Buildroot	108	110	109	**109**
`bs=18000 count=8500 // 145.9 MB`				
➢ komprimace				
○ Beagle Debian	108	110	109	**109**
○ Buildroot	108	111	108	**109**
➢ dekomprimace				
○ OrangePi Debian	118	117	116	**117**
○ Buildroot	105	106	107	**106**

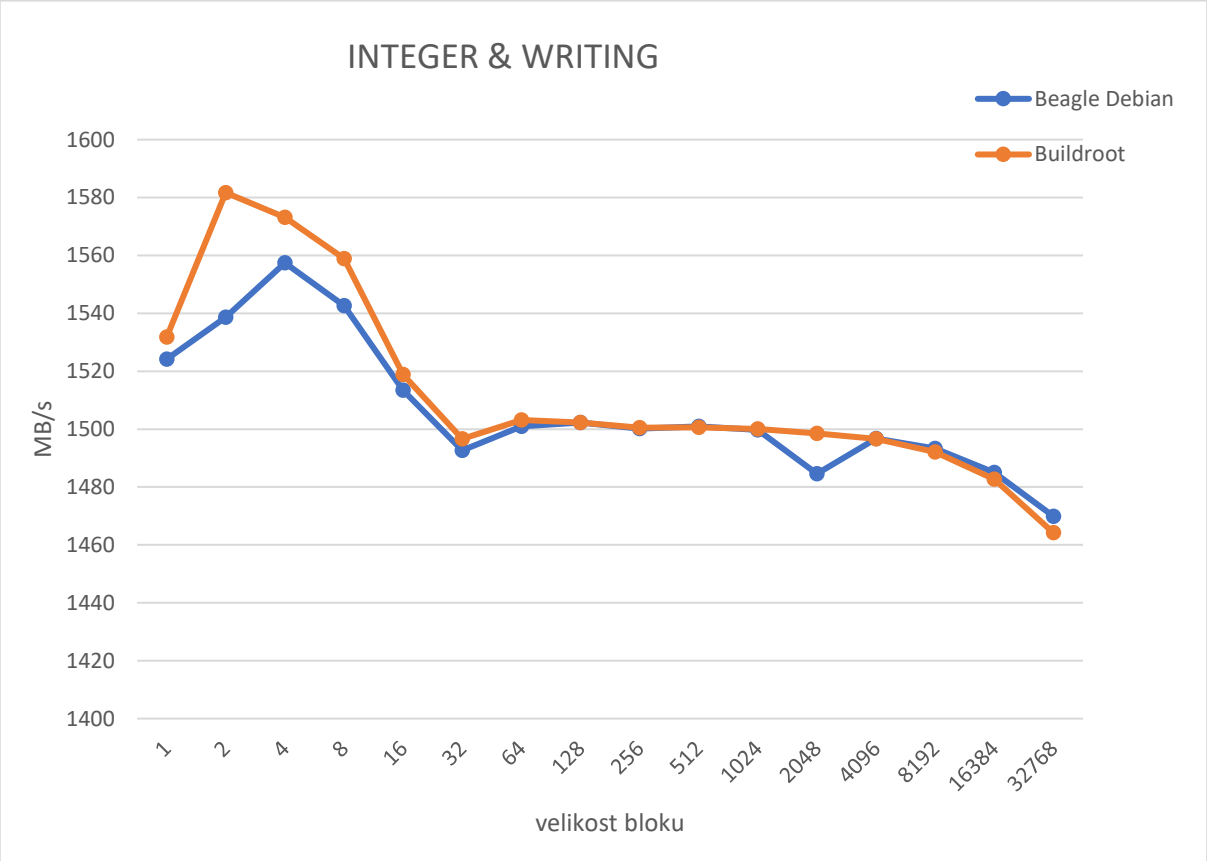


Obrázek 37: Benchmark BeagleBone Black – BW (1)



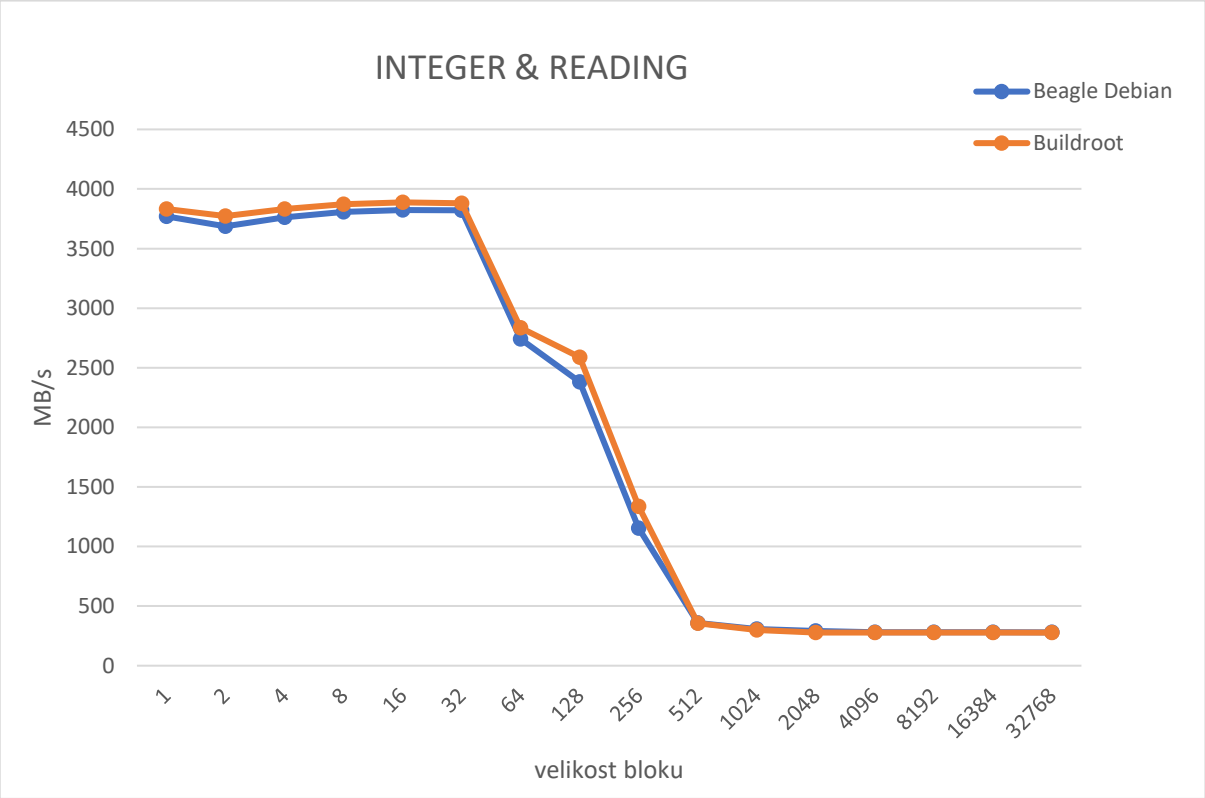
Obrázek 38: Benchmark BeagleBone Black – BW (2)

BeagleBone Black - RAMSPEED - INTEGER & WRITING MB/s		
velikost bloku (kb)	Beagle Debian	Buildroot
1	1524.17	1531.74
2	1538.65	1581.69
4	1557.41	1573.2
8	1542.61	1558.87
16	1513.44	1518.91
32	1492.62	1496.64
64	1500.9	1503.13
128	1502.26	1502.23
256	1500.16	1500.52
512	1500.97	1500.63
1024	1499.72	1500.08
2048	1484.57	1498.52
4096	1496.74	1496.62
8192	1493.31	1492.05
16384	1484.98	1482.69
32768	1469.89	1464.19



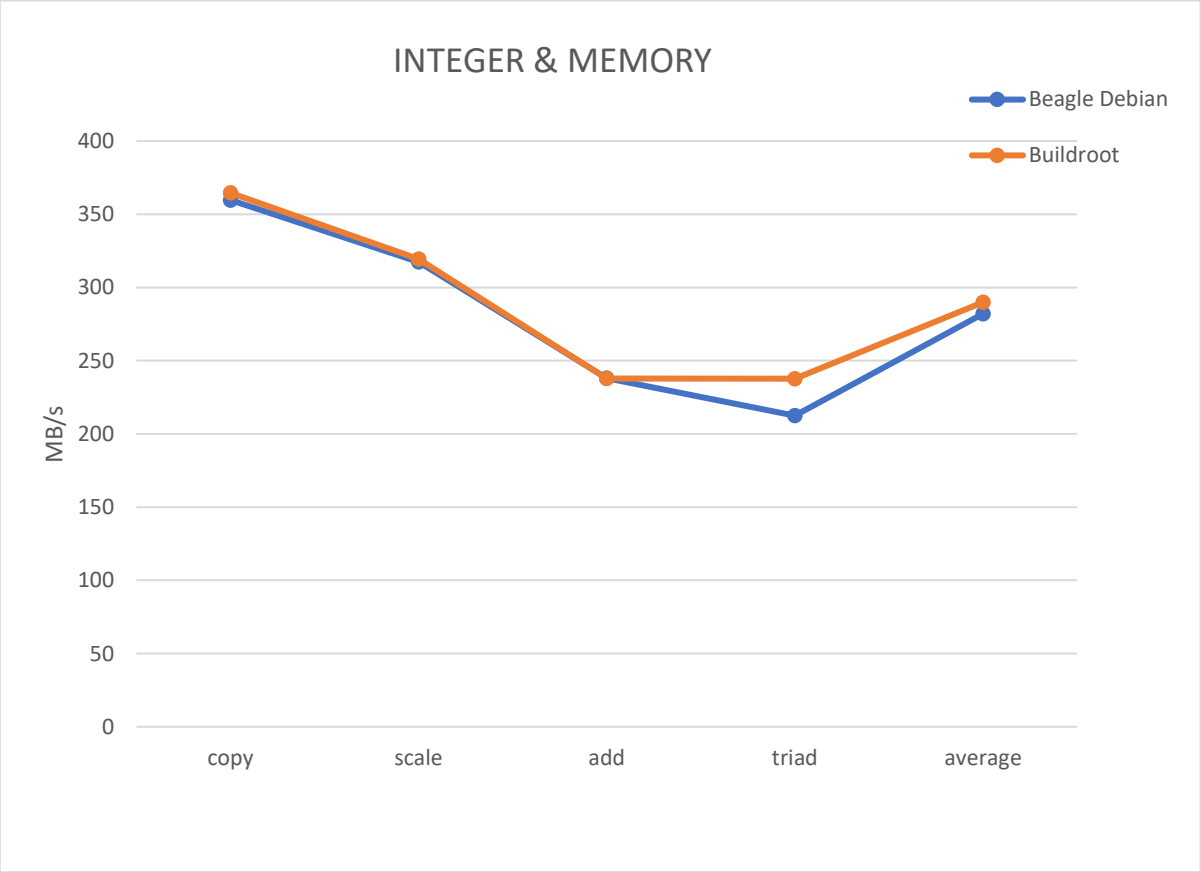
Obrázek 39: Benchmark BeagleBone Black – Integer writing

BeagleBone Black - RAMSPEED - INTEGER & READING MB/s		
velikost bloku (kb)	Beagle Debian	Buildroot
1	3770.81	3833.09
2	3686.75	3772.38
4	3761.58	3833.04
8	3806.95	3871.22
16	3823.21	3887.49
32	3820.26	3881.24
64	2742.18	2834.4
128	2380.57	2587
256	1154.85	1335.14
512	359.35	355.83
1024	307.76	299.74
2048	290.87	279.34
4096	281.38	277.51
8192	279.7	277.4
16384	279.73	277.41
32768	279.59	277.38



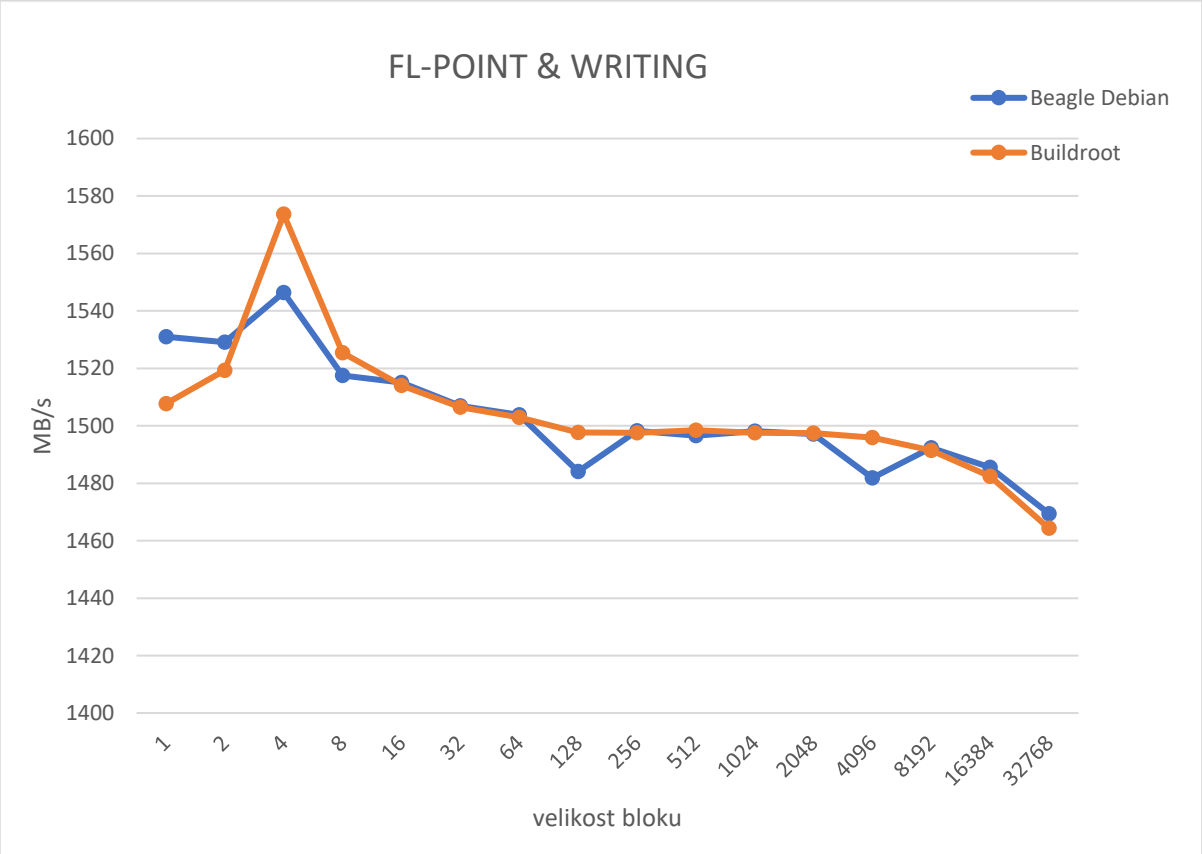
Obrázek 40: Benchmark BeagleBone Black – Integer reading

BeagleBone Black - RAMSPEED - INTEGER & MEMORY MB/s		
	Beagle Debian	Buildroot
copy	359.72	364.61
scale	317.45	319.28
add	237.96	237.8
triad	212.47	237.71
average	281.9	289.85



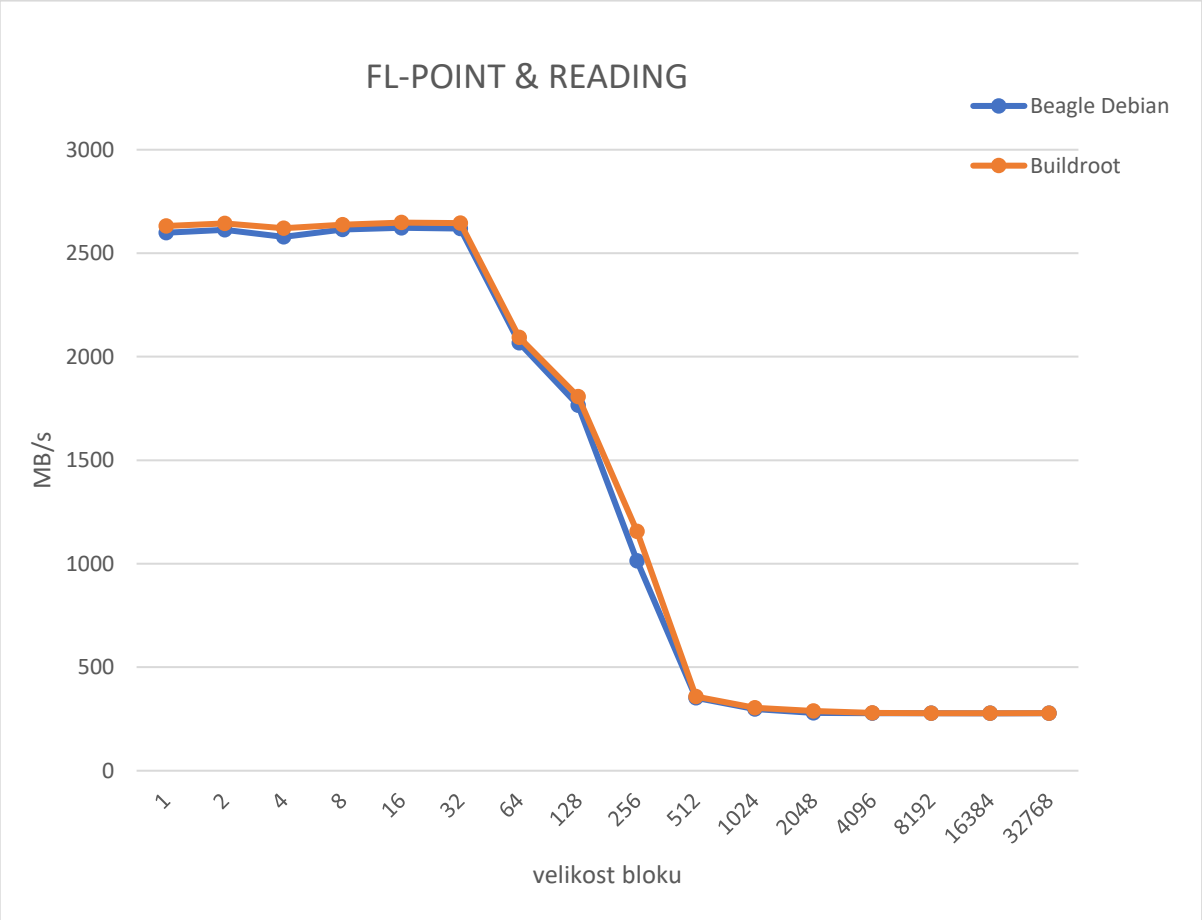
Obrázek 41: Benchmark BeagleBone Black – Integer memory

BeagleBone Black - RAMSPEED – FL-POINT & WRITING MB/s		
velikost bloku (kb)	Beagle Debian	Buildroot
1	1531.01	1507.66
2	1529.16	1519.31
4	1546.43	1573.64
8	1517.51	1525.4
16	1515.02	1514.11
32	1506.98	1506.46
64	1503.77	1502.91
128	1484.08	1497.68
256	1498.18	1497.53
512	1496.56	1498.44
1024	1498.07	1497.54
2048	1497.17	1497.49
4096	1481.8	1495.87
8192	1492.3	1491.49
16384	1485.58	1482.42
32768	1469.41	1464.3



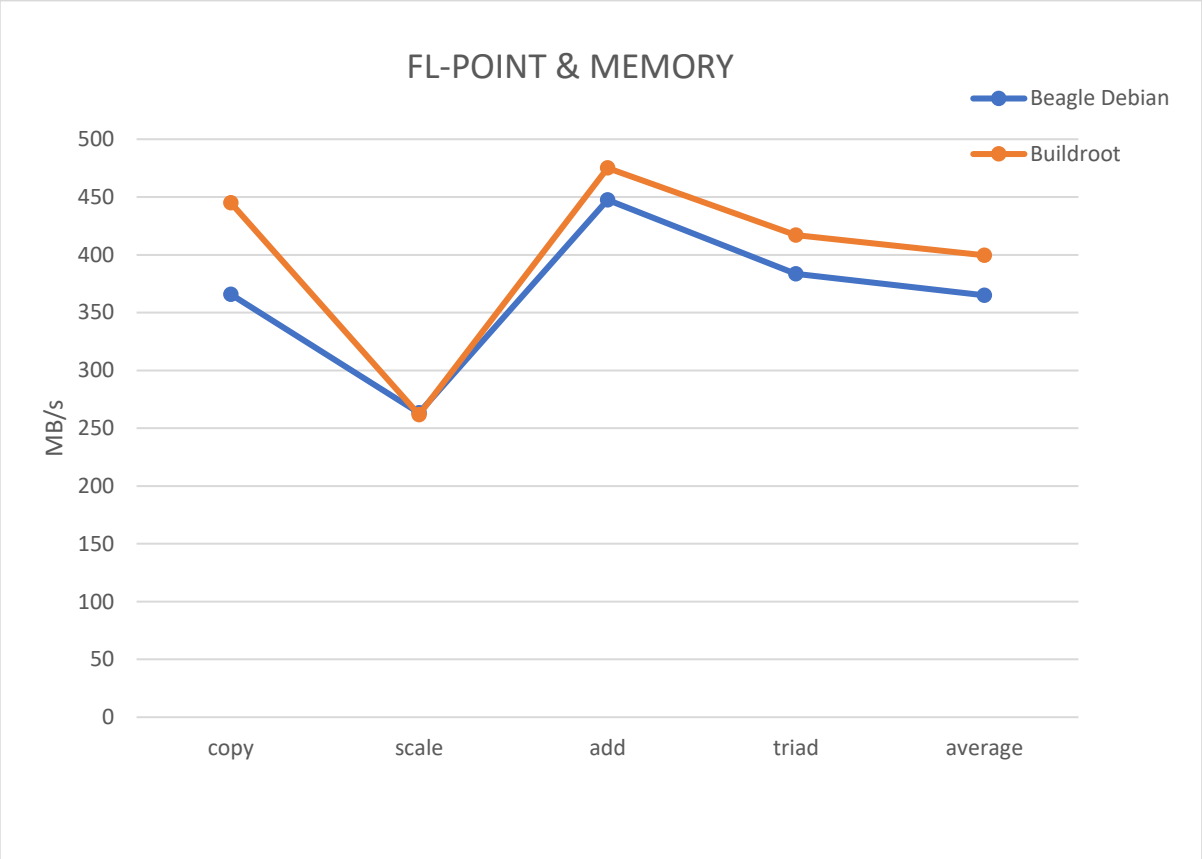
Obrázek 42: Benchmark BeagleBone Black – Fl-point writing

BeagleBone Black - RAMSPEED – FL-POINT & READING MB/s		
velikost bloku (kb)	Beagle Debian	Buildroot
1	2598.56	2631.6
2	2612.49	2643.35
4	2578.88	2620.26
8	2614.45	2638.49
16	2622.91	2647.88
32	2619.15	2645.07
64	2066.41	2093.21
128	1765.03	1806.69
256	1013.42	1155.57
512	351.3	357.73
1024	297.14	303.97
2048	279.27	289.19
4096	277.12	279.22
8192	277.12	278.15
16384	277.04	278.19
32768	277.09	278.12



Obrázek 43: Benchamrk BeagleBone Black – Fl-point reading

BeagleBone Black - RAMSPEED – FL-POINT & MEMORY MB/s		
velikost bloku (kb)	Beagle Debian	Buildroot
copy	365.79	445.09
scale	263.23	261.71
add	447.52	475.08
triad	383.64	417.09
average	365.04	399.74



Obrázek 44: Benchmark BeagleBone Black – Fl-point memory

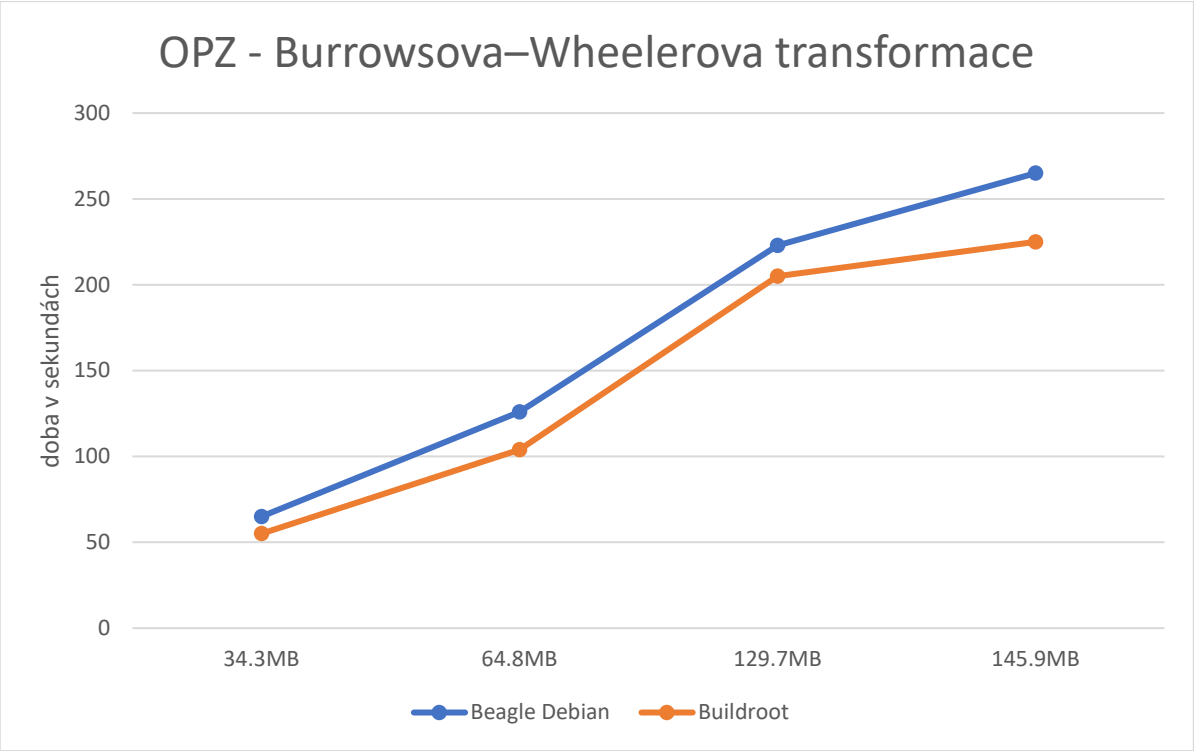
10.3 Benchmark – Orange Pi Zero

Burrowsova–Wheelerova transformace – komprimace / dekomprimace

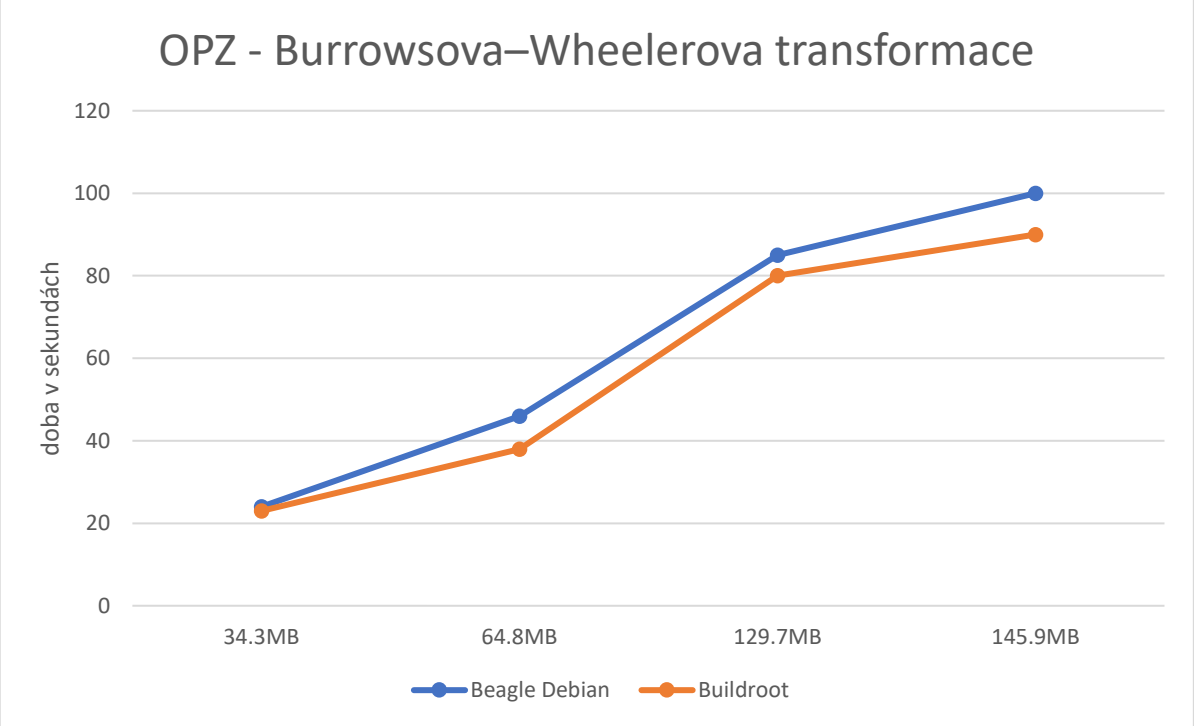
soubor vytvořen pomocí nástroje **dd** o velikosti *count*=**počet** *bs*=**náhodných_bajtů**

```
dd if=/dev/random of=myrandom bs=x count=y
```

	měřené hodnoty			sum	~
`bs=8000 count=4500 // 34.3 MB`					
➢ komprimace					
○ OrangePi Debian	63	65	67	195	**65**
○ Buildroot	57	54	54	165	**55**
➢ dekomprimace					
○ OrangePi Debian	22	25	25	72	**24**
○ Buildroot	21	24	24	69	**23**
`bs=8000 count=8500 // 64.8 MB`					
➢ komprimace					
○ OrangePi Debian	125	123	130	378	**126**
○ Buildroot	104	102	106	312	**104**
➢ dekomprimace					
○ OrangePi Debian	43	46	49	138	**46**
○ Buildroot	39	37	38	114	**38**
`bs=8000 count=17000 // 129.7 MB`					
➢ komprimace					
○ OrangePi Debian	223	225	221	669	**223**
○ Buildroot	203	206	206	615	**205**
➢ dekomprimace					
○ OrangePi Debian	84	86	85	255	**85**
○ Buildroot	81	79	80	240	**80**
`bs=18000 count=8500 // 145.9 MB`					
➢ komprimace					
○ OrangePi Debian	265	270	260	795	**265**
○ Buildroot	226	224	225	675	**225**
➢ dekomprimace					
○ OrangePi Debian	98	101	101	300	**100**
○ Buildroot	89	90	91	270	**90**

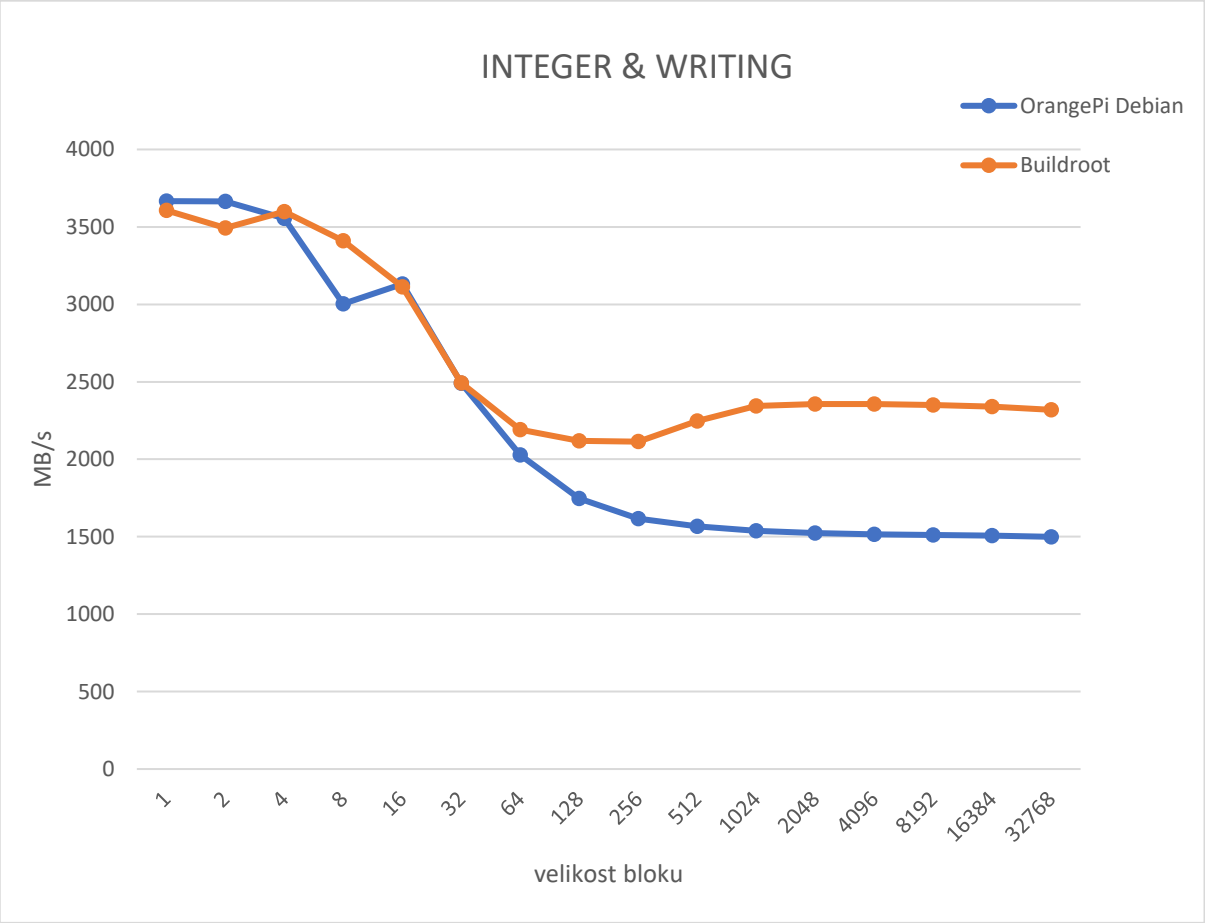


Obrázek 45: Benchmark Orange Pi Zero – BW (1)



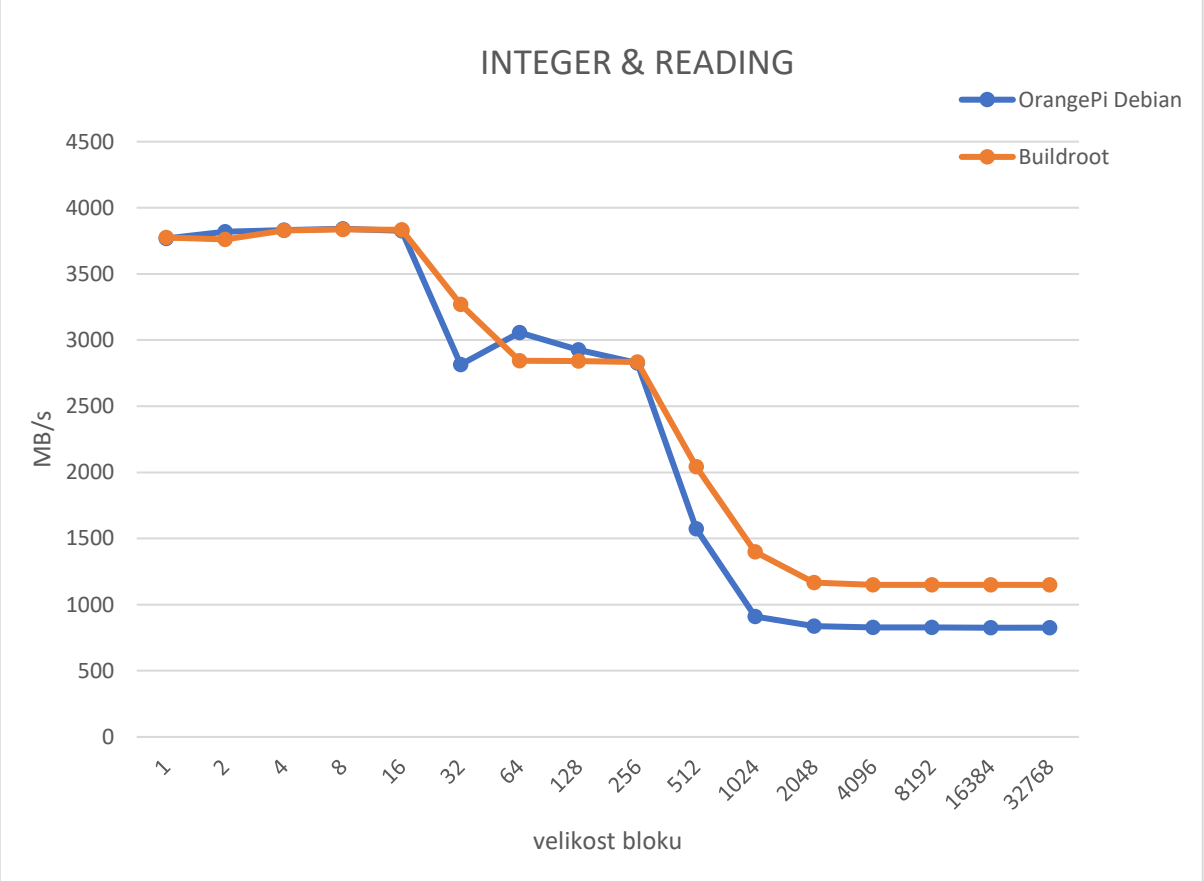
Obrázek 46: Benchmark Orange Pi Zero – BW (2)

Orange Pi Zero - RAMSPEED - INTEGER & WRITING MB/s		
velikost bloku (kb)	OrangePi Debian	Buildroot
1	3666.39	3607.11
2	3664.03	3492.58
4	3555.99	3599.01
8	3003.52	3411.36
16	3131.07	3113.63
32	2491.23	2492.38
64	2026.85	2191.07
128	1745.82	2118.41
256	1616.12	2113.66
512	1567.77	2247.62
1024	1537.46	2343.03
2048	1523.7	2356.23
4096	1515.35	2355.85
8192	1511.74	2349.93
16384	1506.43	2339.53
32768	1499.21	2319.91



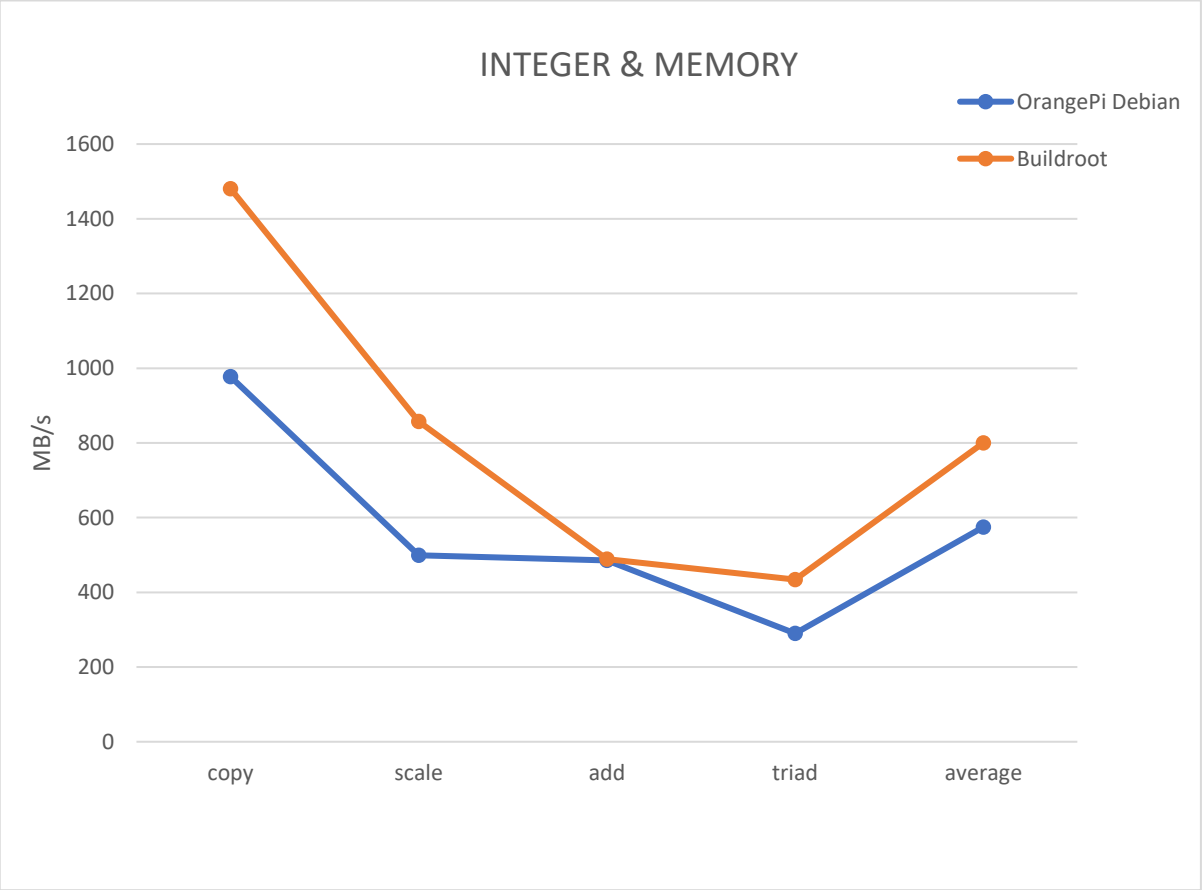
Obrázek 47: Benchmark Orange Pi Zero – Integer writing

Orange Pi Zero - RAMSPEED - INTEGER & READING MB/s		
velikost bloku (kb)	OrangePi Debian	Buildroot
1	3768.76	3775.06
2	3818	3761.07
4	3831.9	3827.9
8	3841.4	3836.95
16	3825.04	3832.86
32	2814	3268.46
64	3057.67	2843.16
128	2925.28	2841.19
256	2827.76	2834.25
512	1573.24	2041.53
1024	910.19	1398.98
2048	836.28	1165.88
4096	827.43	1149.98
8192	826.57	1149.5
16384	825.89	1149.33
32768	825.83	1149.3



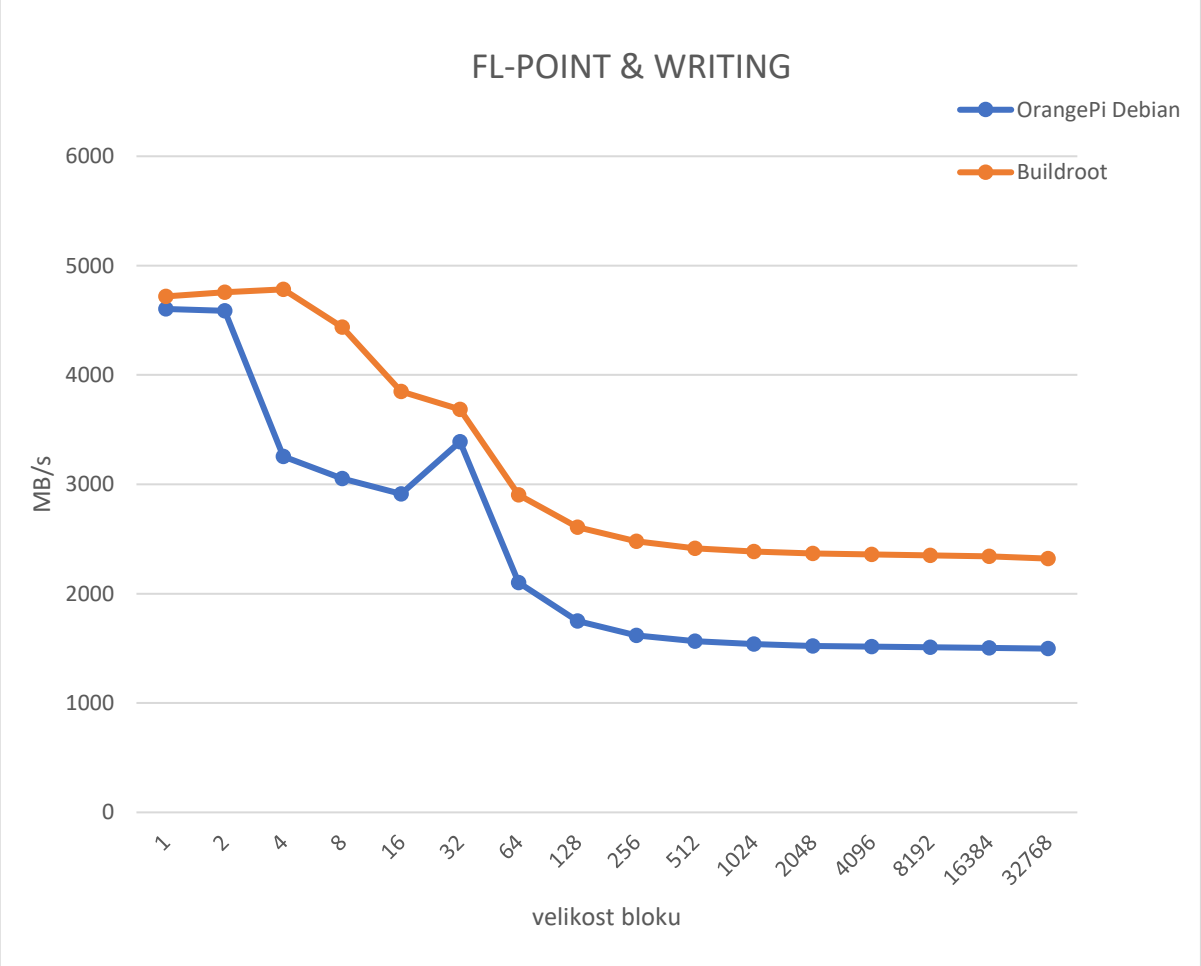
Obrázek 48: Benchmark Orange Pi Zero – Integer reading

Orange Pi Zero - RAMSPEED - INTEGER & MEMORY MB/s		
	OrangePi Debian	Buildroot
copy	977.24	1480.76
scale	499.16	857.55
add	485.76	488.63
triad	289.94	434.1
average	574.28	799.76



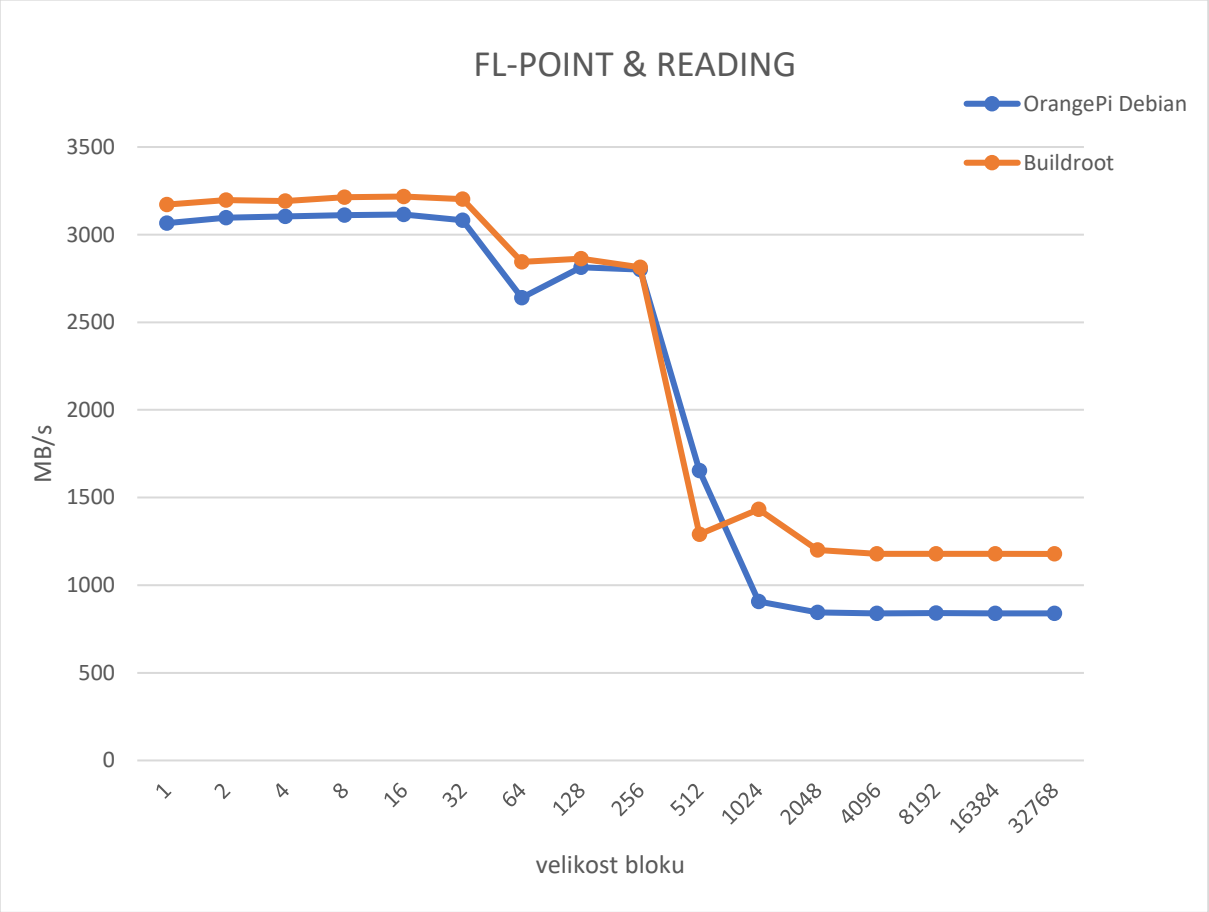
Obrázek 49: Benchmark Orange Pi Zero – Integer memory

Orange Pi Zero - RAMSPEED – FL-POINT & WRITING MB/s		
velikost bloku (kb)	OrangePi Debian	Buildroot
1	4603.43	4718.44
2	4586.84	4756.78
4	3255.07	4782.85
8	3053.92	4438.08
16	2912.55	3848.36
32	3389.76	3684.83
64	2102.31	2903.81
128	1749.84	2608.42
256	1619.67	2479.24
512	1566.35	2414.8
1024	1539.07	2384.74
2048	1520.61	2368.73
4096	1516.83	2359.76
8192	1511.08	2351.31
16384	1504.53	2340.11
32768	1497.56	2320.73



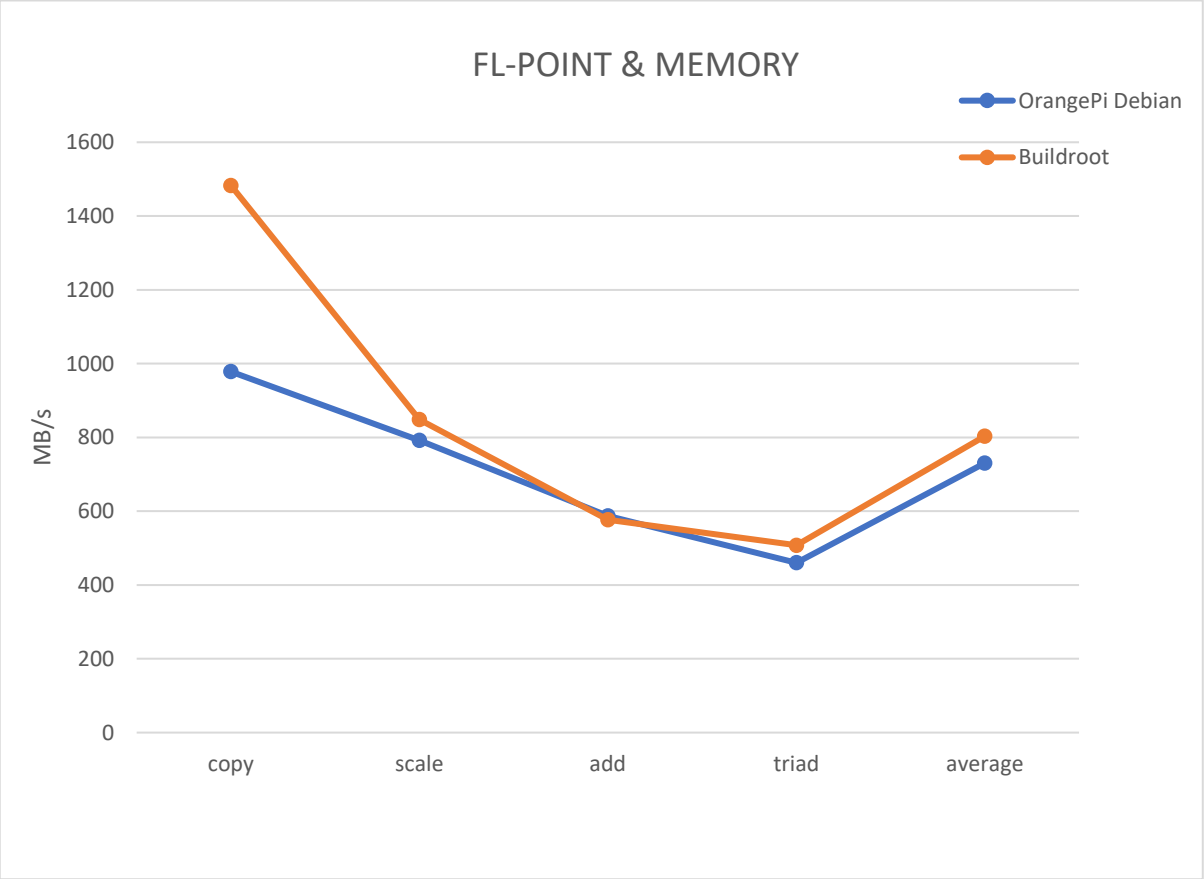
Obrázek 50: Benchmark Orange Pi Zero – Integer writing

Orange Pi Zero - RAMSPEED – FL-POINT & READING MB/s		
velikost bloku (kb)	OrangePi Debian	Buildroot
1	3065.16	3171.81
2	3096.21	3196.32
4	3103.19	3191.33
8	3110.74	3212.79
16	3114.87	3217.46
32	3082.85	3202.08
64	2640.22	2845
128	2813.95	2863.02
256	2801.35	2814.38
512	1653.83	1290.53
1024	906.51	1432.57
2048	843.95	1200.8
4096	839.03	1179.24
8192	841.2	1178.43
16384	839.89	1178.34
32768	839.2	1178.28



Obrázek 51: Benchmark Orange Pi Zero – Fl-point reading

Orange Pi Zero - RAMSPEED – FL-POINT & MEMORY MB/s		
	OrangePi Debian	Buildroot
copy	978.32	1482.99
scale	791.77	848.21
add	587.63	577.05
triad	460.25	507.88
average	730.51	803.02



Obrázek 52: Benchmark Orange Pi Zero – Fl-point memory

10.4 Náhled konfiguračního souboru – Raspberry Pi 3

```
BR2_cortex_a53=y
BR2_ARM_FPU_NEON_VFPV4=y
BR2_CCACHE=y
BR2_PACKAGE_HOST_LINUX_HEADERS_CUSTOM_4_19=y
BR2_TARGET_GENERIC_ISSUE="diplomova_prace_vojta_hrdina_rp3"
BR2_ROOTFS_DEVICE_CREATION_DYNAMIC_EUDEV=y
BR2_SYSTEM_DHCP="eth0"
BR2_ROOTFS_OVERLAY="board/dpvh/raspberrypi3/rootfs_overlay"
BR2_ROOTFS_POST_IMAGE_SCRIPT="board/dpvh/raspberrypi3/post-image.sh"
BR2_LINUX_KERNEL_CUSTOM_GIT=y
BR2_LINUX_KERNEL_CUSTOM_REPO_URL="https://github.com/raspberrypi/linux.git"
BR2_LINUX_KERNEL_CUSTOM_REPO_VERSION="rpi-4.19.y"
BR2_LINUX_KERNEL_DEFCONFIG="bcm2709"
BR2_LINUX_KERNEL_INTREE_DTS_NAME="bcm2710-rpi-3 bcm2710-rpi-cm3"
BR2_PACKAGE_LINUX_TOOLS_GPIO=y
BR2_PACKAGE_B43_FIRMWARE=y
BR2_PACKAGE_LINUX_FIRMWARE=y
BR2_PACKAGE_RPI_FIRMWARE=y
BR2_PACKAGE_RPI_WIFI_FIRMWARE=y
BR2_PACKAGE_LINUXCONSOLETOOLS=y
BR2_PACKAGE_LSHW=y
BR2_PACKAGE_MINICOM=y
BR2_PACKAGE_PIGPIO=y
BR2_PACKAGE_RASPI_GPIO=y
BR2_PACKAGE_SETSERIAL=y
BR2_PACKAGE_NCURSES_TARGET_PROGS=y
BR2_PACKAGE_CONNMAN=y
BR2_PACKAGE_IPERF3=y
BR2_PACKAGE_IPROUTE2=y
BR2_PACKAGE_IW=y
BR2_PACKAGE_NTP=y
BR2_PACKAGE_OPENSSH=y
BR2_PACKAGE_WPA_SUPPLICANT_AP_SUPPORT=y
BR2_PACKAGE_HTOP=y
BR2_PACKAGE_UTIL_LINUX_LIBUUID=y
BR2_PACKAGE_MC=y
BR2_TARGET_ROOTFS_EXT2_SIZE="200M"
BR2_PACKAGE_HOST_GENIMAGE=y
```

10.5 Náhled konfiguračního souboru – BeagleBone Black

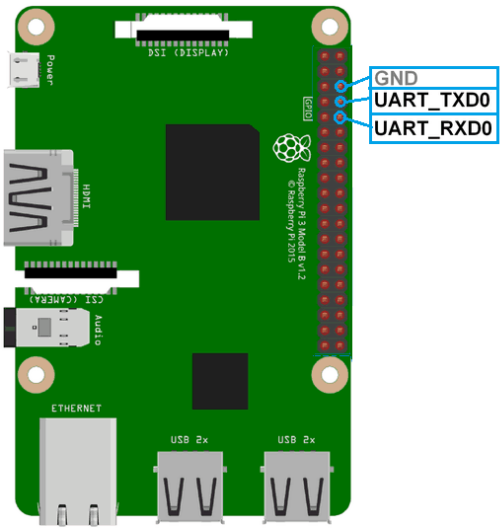
```
BR2_cortex_a8=y
BR2_CCACHE=y
BR2_GLOBAL_PATCH_DIR="board/beaglebone/patches"
BR2_PACKAGE_HOST_LINUX_HEADERS_CUSTOM_4_19=y
BR2_TOOLCHAIN_BUILDROOT_WCHAR=y
BR2_TARGET_GENERIC_HOSTNAME="dpvh_bbb"
BR2_SYSTEM_DHCP="eth0"
BR2_ROOTFS_POST_BUILD_SCRIPT="board/beaglebone/post-build.sh"
BR2_ROOTFS_POST_IMAGE_SCRIPT="support/scripts/genimage.sh"
BR2_ROOTFS_POST_SCRIPT_ARGS="-c board/beaglebone/genimage.cfg"
BR2_LINUX_KERNEL_CUSTOM_TARBALL=y
BR2_LINUX_KERNEL_CUSTOM_TARBALL_LOCATION="$(call
github,beagleboard,linux,4.19.79-ti-r30)/linux-4.19.79-ti-r30.tar.gz"
BR2_LINUX_KERNEL_DEFCONFIG="omap2plus"
BR2_LINUX_KERNEL_DTS_SUPPORT=y
BR2_LINUX_KERNEL_INTREE_DTS_NAME="am335x-boneblack"
BR2_LINUX_KERNEL_NEEDS_HOST_OPENSSL=y
BR2_PACKAGE_GETTEXT=y
BR2_PACKAGE_E2FSPROGS=y
BR2_PACKAGE_LINUX_FIRMWARE=y
BR2_PACKAGE_WILINK_BT_FIRMWARE=y
BR2_PACKAGE_SETSERIAL=y
BR2_PACKAGE_IPROUTE2=y
BR2_PACKAGE_IPTABLES_NFTABLES=y
BR2_PACKAGE_IPUTILS=y
BR2_TARGET_ROOTFS_EXT2_4=y
BR2_TARGET_UBOOT_CUSTOM_VERSION_VALUE="2020.04"
BR2_TARGET_UBOOT_BOARD_DEFCONFIG="am335x_evm"
BR2_TARGET_UBOOT_NEEDS_DTC=y
BR2_TARGET_UBOOT_FORMAT_IMG=y
BR2_TARGET_UBOOT_SPL=y
BR2_TARGET_UBOOT_SPL_NAME="MLO"
BR2_PACKAGE_HOST_DOSFSTOOLS=y
BR2_PACKAGE_HOST_GENIMAGE=y
BR2_PACKAGE_HOST_MTOOLS=y
BR2_PACKAGE_HOST_PYTHON3=y
BR2_PACKAGE_HOST_UBOOT_TOOLS=y
BR2_PACKAGE_HOST_UBOOT_TOOLS_FIT_SUPPORT=y
```

10.6 Náhled konfiguračního souboru – Orange Pi Zero

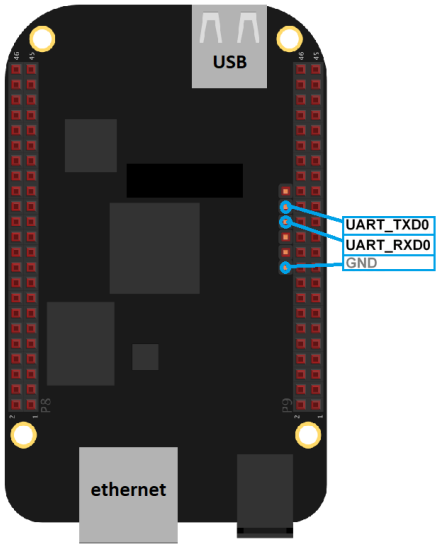
```
BR2_cortex_a7=y
BR2_ARM_FPU_VFPV4=y
```

```
BR2_CCACHE=y
BR2_GLOBAL_PATCH_DIR="board/orangepi/orangepi-zero/patches"
BR2_PACKAGE_HOST_LINUX_HEADERS_CUSTOM_5_3=y
BR2_TARGET_GENERIC_HOSTNAME="OrangePi_Zero"
BR2_TARGET_GENERIC_ISSUE="Welcome to Buildroot for the Orange Pi Zero"
BR2_ROOTFS_DEVICE_CREATION_DYNAMIC_MDEV=y
BR2_ROOTFS_POST_IMAGE_SCRIPT="support/scripts/genimage.sh"
BR2_ROOTFS_POST_SCRIPT_ARGS="-c board/orangepi/orangepi-zero/genimage.cfg"
BR2_LINUX_KERNEL_CUSTOM_VERSION_VALUE="5.3.8"
BR2_LINUX_KERNEL_DEFCONFIG="sunxi"
BR2_LINUX_KERNEL_CONFIG_FRAGMENT_FILES="board/orangepi-zero/linux-extras.config"
BR2_LINUX_KERNEL_INTREE_DTS_NAME="sun8i-h2-plus-orangepi-zero"
BR2_PACKAGE_ARMBIAN_FIRMWARE_XR819=y
BR2_PACKAGE_XR819_XRADIO=y
BR2_PACKAGE_CONNMAN=y
BR2_PACKAGE_IPROUTE2=y
BR2_PACKAGE_IW=y
BR2_PACKAGE_WPA_SUPPLICANT_AP_SUPPORT=y
BR2_PACKAGE_HTOP=y
BR2_TARGET_ROOTFS_EXT2_4=y
BR2_TARGET_UBOOT=y
BR2_TARGET_UBOOT_BUILD_SYSTEM_KCONFIG=y
BR2_TARGET_UBOOT_CUSTOM_VERSION=y
BR2_TARGET_UBOOT_CUSTOM_VERSION_VALUE="2019.10"
BR2_TARGET_UBOOT_BOARD_DEFCONFIG="orangepi_zero"
BR2_TARGET_UBOOT_NEEDS_DTC=y
BR2_TARGET_UBOOT_NEEDS_PYLIBFDT=y
BR2_TARGET_UBOOT_FORMAT_CUSTOM=y
BR2_TARGET_UBOOT_FORMAT_CUSTOM_NAME="u-boot-sunxi-with-spl.bin"
BR2_TARGET_UBOOT_BOOT_SCRIPT_SOURCE="board/orangepi/orangepi-zero/boot.cmd"
BR2_PACKAGE_HOST_UBOOT_TOOLS=y
```

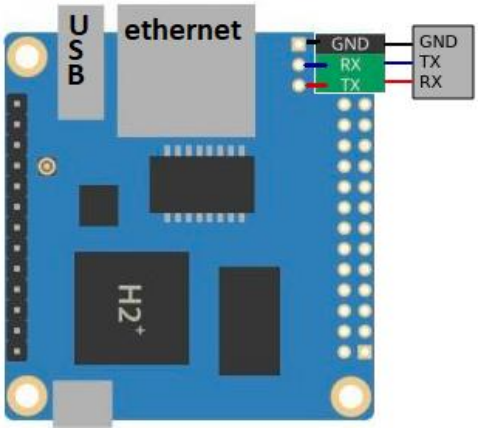
10.7 Sériová rozhraní zařízení



Obrázek 53: Raspberry Pi 3 - sériové rozhraní



Obrázek 54: BeagleBone Black – sériové rozhraní



Obrázek 55: Orange Pi Zero – sériové rozhraní

ZDROJE

[1] MAUERER, Wolfgang. Professional Linux kernel architecture. Indianapolis, IN: Wiley Pub., c2008. ISBN 978-0470343432.

[2] OpenADK - Open Source Appliance Development Kit [online]. [cit. 2018-10-08]. Dostupné z: https://openadk.org/.

[3] Buildroot: Making Embedded Linux Easy [online]. [cit. 2018-10-08]. Dostupné z: https://buildroot.org/.

[4] Raspberry Pi Documentation. *Raspberrypi.org* [online]. Raspberry Pi Foundation [cit. 2020-05-18]. Dostupné z: www.raspberrypi.org/documentation/

[5] BeagleBone Black. *Beagle Board* [online]. [cit. 2020-05-18]. Dostupné z: beagleboard.org/black

[6] *Orangepi.org/orangepizero/* [online]. [cit. 2020-05-18]. Dostupné z: www.orangepi.org/orangepizero/

[7] MASTERS, Jon a Richard BLUM. *Linux profesionálně: programování aplikací*. Brno: Zoner Press, 2008. Encyklopedie Zoner Press. ISBN 978-80-86815-71-8.

[8] *Wikipedia The Free Encyclopedia* [online]. [cit. 2020-05-19]. Dostupné z: wikipedia.org/wiki/GNU_binutils

[9] *From Wikipedia, the free encyclopedia* [online]. [cit. 2020-05-20]. Dostupné z: en.wikipedia.org/wiki/BusyBox

[10] *BUSYBOX* [online]. [cit. 2020-05-20]. Dostupné z: busybox.net/

[11] *Barebox* [online]. [cit. 2020-05-20]. Dostupné z: barebox.org/

[12] *Qemu* [online]. [cit. 2020-05-20]. Dostupné z: www.qemu.org/

[13] *Balena* [online]. [cit. 2020-05-20]. Dostupné z: www.balena.io/etcher/

[14] BeagleBone Black. *Digikey.com* [online]. [cit. 2020-05-21]. Dostupné z: www.digikey.com/eewiki/display/linuxonarm/BeagleBone+Black

[15] Úvod do embedded systémů. *DPS Elektronika od A do Z* [online]. 2020 [cit. 2020-07-10]. Dostupné z: https://www.dps-az.cz/vyvoj/id:6173/uvod-do-embedded-systemu

[16] COVEX PŘIBYL, ADAM. ARM - Linux není jen x86. *Root.cz* [online]. 2005 [cit. 2020-07-10]. Dostupné z: https://www.root.cz/clanky/arm-linux-neni-jen-x86/

[17] Toolchains. *Elinux.org* [online]. 2007 [cit. 2020-08-10]. Dostupné z: https://elinux.org/Toolchains