Portovanje operativnog sistema Tizen za arhitekturu MIPS32

Dragan Čečavac, Dejan Latinović i Petar Jovanović

***Apstrakt*—U ovom radu je opisan postupak portovanja Tizen 2.3 wearable operativnog sistema za 32-bitnu MIPS procesorsku platformu. Tizen je operativni sistem namenjen širokom spektru uređaja koji promoviše ideju njihove integracije u jedinstven ekosistem koji pruža konzistentno korisničko iskustvo. MIPS predstavlja RISC procesorsku arhitekturu koja uz nisku potrošnju energije ostvaruje dobre performanse, sa posebno dominantnom pozicijom u sferi mrežnih uređaja. U radu su dati detaljniji opisi operativnog sistema i platforme, predstavljeni su razlozi koji su doveli do portovanja, opisane su faze rada i prikazani rezultati verifikacionih testova.**

***Ključne reči*—MIPS; Pametni sat; *Wearable*; *Internet of Things*.**

I. Uvod

Tizen je fleksibilan operativni sistem otvorenog koda kreiran da zadovolji potrebe svih aktera u ekosistemu mobilnih i povezanih uređaja, uključujući proizvođače uređaja, mobilne operatere, programere aplikacija i nezavisne softverske prodavce (*ISV*). Razvijen je u okviru Linux fondacije, baziran na Linux kernelu i GNU C biblioteci. Aktivnosti vezane za ravoj kontroliše *Technical Steering Group* (*TSG*) koji predvode kompanije Samsung Electronics i Intel [1]. Kompletan razvoj je tekao u programerskoj zajednici po principima razvoja otvorenog koda i svima koji su zainteresovani je omogućeno da se uključe u dalji razvoj.

Tizen dolazi u više profila sa ciljem da zadovolji široke potrebe industrije poput pametnih telefona, pametnih satova, tableta, računara, televizora, kamera, biomedicinskih uređaja, vozila, bankarskih sistema itd.

Pored toga što je Tizen zamišljen kao operativni sistem svega, on je takođe podrazumevana platforma kompanije Samsung Electronics, koja podržava njenu viziju interneta stvari (*IoT*). U osnovi je ideja da svi uređaju koje korisnik poseduje budu povezani i u međusobnoj komunikaciji, uključujući i uređaje na kojima se ne očekuje prisustvo napredne tehnologije [2].

MIPS32 arhitektura je vlasništvo Imagination Technologies Group plc. Ima dobre performanse i visoku efikasnost uz nisku potrošnju energije. Zauzima jaku poziciju u domenu *home entertainment*, *embedded* i mrežnih uređaja, sa zabeleženim rastom u domenu mobilnih, *wearable* i *IoT* uređaja. MIPS procesori se nalaze u milijardama uređaja širom sveta. Ova arhitektura pruža robustan skup instrukcija, skalabilnost 32-bitne arhitekture prema 64-bitnoj i rasprostranjenu podršku partnera i korisnika licenci. Arhitektura je podržana u širokom ekosistemu operativnih sistema, alata, komercijalnog i *open source* softvera [3, 4].

Svim procesorskim arhitekturama je u interesu da podržavaju što širi spektar operativnih sistema. Posebno kada su u pitanju ambiciozni projekti kao što je Tizen, koji uz podršku tehnoloških lidera koji iza njega stoje može mnogo da postigne i nametne se kao nezaobilazni deo naše svakodnevice. Tizen je izvorno razvijen sa podrškom za ARM i Intel procesorske arhitekture, a imajući u vidu visoki potencijal u poslovnom segmentu na koji se MIPS arhitektura širi, veoma bitno je bilo da se obezbedi njegova podrška za MIPS.

Pre kreiranja Tizen podrške, za MIPS arhitekturu su prilagođeni i razni drugi operativni sistemi. Iz segmenta mobilnih uređaja, to su Android [5, 6] i Firefox OS [7]. Iz segmenta operativnih sistema za standardne računare primeri su Linux [8] i RT-preemtp [9]. A postoji i veliki broj pratećih alata [10].

Budući da je već postojala MIPS podrška u više srodnih oblasti, zaključeno je da portovanje Tizen operativnog sistema ne bi zahtevalo značajne resurse te da su benefiti koji proizilaze iz portovanja za MIPS32 procesorsku arhitekturu obostrani.

Rad se pored prvog uvodnog, sastoji iz još pet poglavlja. Drugo poglavlje pruža informacije vezane za sam proces prevođenja koda, kao i informacije o savladanim preprekama. Treće poglavlje opisuje postupak podizanja Tizen operativnog sistema na fizičkom uređaju, uz kratko osvrtanje na najbitnije probleme. U četvrtom poglavlju su prikazani rezultati verifikacije finalnog rešenja. Peto poglavlje opisuje javne prezentacije finalnog rešenja sa opisom razvijenog mehanizma za automatsko instaliranje aplikacija prilikom pokretanja, dok je u šestom poglavlju predstavljen zaključak.

II. Portovanje Tizen operativnog sistema

U procesu portovanja bilo je potrebno proći kroz sledeće faze:

- Podešavanje razvojnog okruženja
- Preuzimanje koda
- Sprovođenje analize koda specifičnog za ARM i Intel procesorkse arhitekture
- Dodavanje podrške za MIPS procesorsku

Dragan Čečavac – Istraživačko-razvojni Institut RT-RK, Novi Sad, Srbija (e-mail: dragan.cecavac@rt-rk.com).

Dejan Latinović – Istraživačko-razvojni Institut RT-RK, Novi Sad, Srbija (e-mail: dejan.latinovic@rt-rk.com).

Petar Jovanović – Istraživačko-razvojni Institut RT-RK, Novi Sad, Srbija (e-mail: petar.jovanovic@rt-rk.com).

arhitekturu u delovima gde je detektovana potreba

- Prevođenje inicijalnog skupa paketa
- Inkrementalno prevođenje programskog koda uz sanaciju grešaka po detekciji
- Kreiranje slike sistema

Pre početka portovanja, bilo je potrebno uveriti se da za operativni sistem na kome će biti rađeno portovanje postoji softverska podrška za alate koji će biti korišćeni. Trenutno su podržane samo neke od aktuelnih verzija popularnih Linux distribucija: Ubuntu, openSUSE, Fedora, CentOS i Debian.

Razvojni alati koje treba instalirati su GBS, MIC i qemu-user paket. Prilikom njihove instalacije je potrebno instalirati i sve pakete koji nedostaju, a pomenuti alati ih navode u listi zavisnosti. GBS je alat za razvoj Tizen paketa koji se pokreće iz komandne linije i omogućava lokalno prevođenje Tizen koda [11]. MIC je Tizen alat koji se koristi za kreiranje slike sistema (*system image*) [12]. Slika sistema predstavlja fajl koji sadrži celi fajl sistem, kako je u slučaju Tizena, ili određeni segment fajl sistema, na primer kod Android operativnog sistema. qemu-user paket sadrži različite varijante arhitekturalnih emulatora, a nama je bitan zbog qemu-mipsel emulatora MIPS arhitekture, čija uloga u prevođenju je opisana u daljem tekstu.

Za preuzimanje izvornog koda potrebno je na web stranici na koju pokazuje [13] napraviti nalog za pristup Gerrit servisu i pratiti uputstva za preuzimanje željene verzije i željenog profila.

Trenutno su dostupni profili: Tizen *mobile* – namenjen za mobilne uređaje; Tizen *wearable* – namenjen za pametne satove; Tizen TV – namenjen za televizore; te Tizen IVI (*In Vehicle Infotainment*) – namenjen za vozila.

Tizen kod je sastavljen iz više manjih projekata, koji su organizovani kao Git repozitorijumi. Neki od njih predstavljaju izmenjene projekte otvorenog koda koji se koriste kao sastavni delovi drugih sistema u zajednici otvorenog koda. Poput projekata Autoconf, BusyBox, X.Org Server, WebKit i mnogih drugih. Po uspešnom prevođenju projekta, kreira se odgovarajući RPM (*Red Hat Package Manager*) paket u poddirektorijumu putanje podešene kao podrazumevani izlazni direktorijum. Putanja se zadaje posle *--buildroot* opcije za GBS alat. U daljem tekstu će često biti korišćen izraz paket, koji ne mora da odgovara izrazu projekat. Jedan projekat može da sadrži kod za jedan ili više paketa.

U fazi analize specifičnog koda za prethodno podržane arhitekture, potrebno je istražiti i razumeti odakle potiču razlike i kako se one odražavaju na MIPS32 arhitekturu. Fajlovi iz raznih projekata, sa najviše arhitekturalno specifičnog koda su bili: naziv-paketa.spec, Makefile.am i CMakeLists.txt. Na njih je bilo potrebno obratiti posebnu pažnju. Neke izmene koje su se odnosile na *flag*-ove programskog prevodioca su bile šablonske i trebalo ih je primeniti u desetinama projekata. Najzastupljenija je bila izmena sledećeg tipa i odnosila se na uklanjanje linker opcije *--hash-style=both*, koja nije bila podržana za MIPS arhitekturu u okviru binutils linkera:

```
diff --git a/packaging/alsa-lib-1.0.25.spec b/packaging/alsa-lib-1.0.25.spec
index c8d6e67..ca5450d 100644
--- a/packaging/alsa-lib-1.0.25.spec
+++ b/packaging/alsa-lib-1.0.25.spec
@@ -43,7 +43,13 @@ ALSA Library package for multimedia framework
middleware development package
 %build
 export CFLAGS+=" -fPIC"
+
+%ifnarch mipsel
 export LDFLAGS+=" -Wl,--warn-unresolved-symbols -Wl,--hash-style=both
-Wl,--as-needed"
+%else
+export LDFLAGS+=" -Wl,--warn-unresolved-symbols -Wl,--as-needed"
+%endif
+
 chmod +x autogen.sh

 %autogen --disable-static
```

Prevođenje svih Tizen paketa može da se zada odjednom, ali u praksi ne treba očekivati da će prevođenje da bude uspešno. Mnogi problemi se uočavaju tek u ovoj fazi. Najjednostavniji pristup je na početku zadati prevođenje svih paketa i implementirati dodatnu podršku za novu arhitekturu tamo gde se uoči potreba. Nakon rešavanja detektovanih problema prevoditi pojedinačno sve pakete kod kojih su postojale greške u prethodnom prevođenju, sve dok se ne reše detektovani problemi, nakon čega se ponovo pokreće prevođenje svih neprevedenih paketa. Pojedini paketi imaju zavisnosti prema drugim paketima i ne mogu se prevoditi dok se paket od koga zavise ne prevede. Imajući ovo u vidu, prevođenje koda se odvija inkrementalno. Sastoji se od prevođenja svih paketa do detektovanja grešaka, zatim pojedinačnog prevođenja paketa nakon otklanjanja grešaka u prevođenju, te ponovnog pristupa prevođenju svih paketa i tako ukrug, do prevođenja svih željenih paketa.

Za prevođenje i linkovanje Tizen koda neophodni su *toolchain* paketi poput GCC, binutils i EGLIBC. *Toolchain* predstavlja skup ulančanih alata za razvoj softvera, gde se izlaz jednog alata prosleđuje na ulaz sledećeg. Može da bude i *cross toolchain* ako izlazni kod nije namenjen za arhitekturu mašine na kojoj se programski kod prevodi [14]. Tizen *toolchain* paketi su deo inicijalnog skupa paketa neophodnih za bilo kakvo prevođenje Tizen koda pomoću GBS. Ovde se javlja problem kokoške i jajeta prilikom prevođenja za nove arhitekture, pošto je za prevođenje *toolchain* paketa potrebno da se već poseduje određena varijanta prevedenih *toolchain* paketa. Do ovih paketa je moguće doći prevođenjem koda bez korišćenja GBS sistema uz pomoć *toolchain*-a koji nije zvanični deo Tizena. Mi smo to postigli prevođenjem inicijalnog skupa paketa na modifikovanom Debian operativnom sistemu pokrenutom na ploči sa MIPS arhitekturom. Ukoliko već postoje prevedeni paketi inicijalnog skupa, najjednostavnije je da se samo preuzmu, kao što se radi u prvom krugu prevođenja Tizen koda za ARM ili Intel arhitekture. Nakon prevođenja ili preuzimanja svih potrebnih paketa, isti treba da se smeste u željeni direktorijum, a putanja do njega se prosleđuje opcijom *--repository* u okviru komande za početak prevođenja.

Nakon što se obezbedi pristup svim paketima iz inicijalnog skupa i nakon što se svi ti paketi ponovo prevedu koristeći GBS, najbolji pristup je izbaciti referencu ka paketima do kojih se došlo nezvaničnim putem i još jednom prevesti sve pakete koji su do tada prevedeni. Na ovaj način se osigurava da ceo Tizen bude preveden odgovarajućim verzijama *toolchain*-a.

Inicijalni skup paketa smo prevodili na ploči sa MIPS arhitekturom, ali sve ostalo, uključujući i drugi krug prevođenja inicijalnog skupa paketa, je prevođeno na mašini sa Intel x64 arhitekturom. Ciljana arhitektura u svim slučajevima je bila MIPS32 revision 2 za *little endian*.

GBS ne koristi *cross toolchain*, nego kreira izolovani *buildroot* u okviru koga se pomoću qemu-mipsel emulira prevođenje koda na ciljnoj mašini. *Toolchain* paketi u ovom slučaju su paketi iz inicijalnog skupa paketa, prevedeni samo za ciljanu arhitekturu.

Prilikom prvog korišćenja GBS alata detektovano je da postoji nekompatibilnost sa MIPS emulatorom. Istraga je utvrdila da nedostaje odgovarajuća *binfmt_misc* maska za mipsel-qemu. *binfmt_misc* predstavlja funkcionalnost Linux kernela koja omogućava pokretanje izvršnih fajlova za procesorske arhitekture nekompatibilne sa radnim sistemom. Pokretanje je omogućeno zadavanjem *binfmt_misc* maske u izvršnom fajlu. Ta maska se poklapa sa maskom u emulatoru, kome se izvršni fajl prosleđuje na izvršavanje bez eksplicitnog navođenja [15]. Za ovaj problem postoji rešenje koje još nije prihvaćeno na *upstream*-u [16], ali problem može da se reši i dopunjavanjem informacija koje su nedostajale pomoću sledećih konzolnih komandi:

```
sudo su
echo ':mipsel-
qemu:M::\x7fELF\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x
00\x08\x00:\xff\xff\xff\xff\xff\xff\xff\x00\xfe\xff\xff\xff\xff\xff\xff\xfe\xf
f\xff\xff:/usr/bin/qemu-mipsel:' > /proc/sys/fs/binfmt_misc/register
```

Sledeći bitan problem u samom sistemu za prevođenje se javio prilikom linkovanja objektnih fajlova u okviru WebKit paketa gde smo se susreli sa problemom nedostatka memorije u qemu-mipsel emulatoru. Problem smo rešili preuzimanjem izvornog koda emulatora i prevođenjem uz povećavanje dostupne memorije. Odnosno, povećavanjem glavnog programskog steka koji se koristi prilikom prevođenja. Izmena koju smo uneli je sledeća:

```
 diff --git a/linux-user/main.c b/linux-user/main.c
index 6599a41..74cf8de 100644
--- a/linux-user/main.c
+++ b/linux-user/main.c
@@ -4099,7 +4099,8 @@ int main(int argc, char **argv, char **envp)
        if (getrlimit(RLIMIT_STACK, &lim) == 0
            && lim.rlim_cur != RLIM_INFINITY
            && lim.rlim_cur == (target_long)lim.rlim_cur) {
-           guest_stack_size = lim.rlim_cur;
+           guest_stack_size = 41943040;
        }
    }
```

Nisu sve izmene implementirane zbog zahteva diktiranih od strane same arhitekture ili problema sa sistemom za prevođenje. Pojedine su bile uzrokovane problemima *toolchain* paketa sa podrškom za MIPS arhitekturu. Najbolji primer je problem sa linkerom binutils zbog koga nam nije bilo dozvoljeno da koristimo opciju *--gc-sections* prilikom prevođenja. Ovaj problem je poznat u zajednici i rešen je u novijim verzijama binutils linkera. Tu podršku nismo mogli da iskoristimo jer bi unapređenje binutils verzije bilo preveliki korak i otežalo kasnije *upstream*-ovanje koda, dok je pomenuta podrška zahtevala značajan napor za integraciju u verziju koja se koristi na Tizen 2.3 wearable. Umesto toga, odlučeno je da izuzmemo ovu opciju linkera (zaduženu za *garbage collection* nekorišćenih ulaznih sekcija) iz desetine projekata gde je korišćena. Izmene vezane za *--gc-sections* opciju su šablonske i njihov oblik je prikazan na primeru izmene za system-resource-resourced paket:

```
diff --git a/CMakeLists.txt b/CMakeLists.txt
index 543cd8b..9c4931f 100644
--- a/CMakeLists.txt
+++ b/CMakeLists.txt
@@ -29,7 +29,13 @@ IF("${CMAKE_BUILD_TYPE}" STREQUAL
"DEBUG")
  SET(VERBOSE 1)
 ELSE()
 #set compile size optimization option in case of none DEBUG
- SET(ADDITIONAL_OFLAGS "-fdata-sections -ffunction-sections -Wl,--
gc-sections -fno-exceptions")
+ FIND_PROGRAM(UNAME NAMES uname)
+ EXEC_PROGRAM("${UNAME}" ARGS "-m" OUTPUT_VARIABLE
"ARCH")
+ IF("${ARCH}" STREQUAL "mips")
+   SET(ADDITIONAL_OFLAGS "-fdata-sections -ffunction-sections -Wl,-
fno-exceptions")
+ ELSE("${ARCH}" STREQUAL "mips")
+   SET(ADDITIONAL_OFLAGS "-fdata-sections -ffunction-sections -Wl,-
-gc-sections -fno-exceptions")
+ ENDIF("${ARCH}" STREQUAL "mips")
  SET(EXTRA_CFLAGS "${EXTRA_CFLAGS}
${ADDITIONAL_OFLAGS}")
 SET(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS}
${ADDITIONAL_OFLAGS}")
 ENDIF()
```

Pojedini paketi su specifični za određenu arhitekturu i nisu predviđeni za portovanje. U ovim slučajevima je potrebno obratiti posebnu pažnju, pošto postoji mogućnost da srodni paketi za različite arhitekture dele jako malo zajedničkog koda ili da ga uopšte ne dele. Ukoliko se zaključi da je posmatrani paket neophodan i za arhitekturu za koju se portuje, tada je potrebno kreirati novi paket i implementirati kompletnu podršku. Paket xorg-x11-misc-mipsel-common nije bio prisutan kao Tizen paket, ali je portovan po uzoru na pakete xorg-x11-arm-common i xorg-x11-i386-common, kako bi se obezbedila zadata X.Org Server funkcionalnost.

Među svim paketima koji se prevode posebno se izdvaja mic-bootstrap. Naime, ovaj paket se prevodi za arhitekturu mašine na kojoj se kod prevodi, a ne za arhitekturu portovanja.

mic-bootstrap paket ne služi da bi se dodao u finalnu sliku sistema, nego se izvršava u okviru MIC alata na mašini na kojoj se prevodio kod i koristi se za kreiranje pomenute slike sistema.

U našem slučaju je za prevođenje je korišćena sledeća komanda:

```
gbs --verbose build --arch=i586 --repository putanja-do-online-repozitorijuma
--buildroot putanja-do-buildroot --threads=1 --skip-conf-repos ./  --include-all
--baselibs
```

Izmena koja dodaje MIPS podršku u ovaj paket je sledeća:

```
diff --git a/packaging/baselibs.conf b/packaging/baselibs.conf
index 77598bc..1301082 100644
--- a/packaging/baselibs.conf
+++ b/packaging/baselibs.conf
@@ -1,4 +1,4 @@
-arch i586 targets i586:x86-arm armv7l:x86-arm
+arch i586 targets i586:x86-arm armv7l:x86-arm mipsel:x86-mips
targettype x86-arm package mic-bootstrap
@@ -7,3 +7,10 @@ targettype x86-arm package mic-bootstrap
    autoreqprov off
    extension -x86-arm
+/
+
+targettype x86-mips package mic-bootstrap
+    targetarch ia64 block!
+    targetarch x86_64 block!
+    autoreqprov off
+    extension -x86-mips
+    +/
```

Svi RPM paketi mogu da se iskoriste prilikom pokretanja operativnog sistema, ali nisu nužno svi potrebni. Na primer, paket bluez nije potreban sistemima koji ne poseduju modul za Bluetooth komunikaciju i samo će nepotrebno da zauzima prostor na disku i da dodatno optereti sistem tokom rada. Generalna preporuka je da se iskoriste svi Tizen specifični paketi koji se koriste prilikom kreiranja slike sistema za ARM i Intel. Oni se smatraju baznim paketima i čine glavnu razliku između Tizena i ostalih Linux distribucija. Paketi poput gdb, strace ili bilo koga drugog *debug* paketa mogu biti korisni u razvojne svrhe, ali ne treba da se dodaju u sliku sistema namenjenu za opštu upotrebu.

Kickstart fajlovi su fajlovi sa .ks ekstenzijom. Koriste se kao ulazni parametar MIC alata i služe za deklarisanje liste paketa koji će da uđu u sliku sistema. Na zvaničnim Tizen repozitorijumima postoje primeri .ks fajlova za ARM [17] i Intel [18], na osnovu kojih mogu da se kreiraju potrebni .ks fajlovi za MIPS. Najbolji način je da se u okviru paketa package-groups u odgovarajućem .yaml fajlu definišu precizne logičke grupe paketa i zatim navedu u .ks fajlu, umesto da se unose liste koje mogu da uključuju i po nekoliko stotina paketa.

III. Portovanje Tizen OS-a za Newton2 platformu

Tizen wearable 2.3 je portovan za Newton2 hardversku platformu. Newton2 proizvodi kompanija Ingenic Semiconductor Co., Ltd. i zasnovana je na Ingenic M200 *SOC*-u, baziranom na MIPS32 revision 2 arhitekturi. Čip sadrži dva procesorka jezgra. Jedno jezgro radi na frekvenciji od 1.2 GHz, dok drugo jezgro, namenjeno za *low power* režim, radi na frekvenciji od 300 MHz . Memoriju čine 4 GB eMMC *flash* i 4 Gbit LPDDR2. Platforma ima mogućnosti Bluetooth i Wi-Fi povezivanja. Sadrži AMOLED displej osetljiv na dodir, dijagonale 1.63 inča, sa rezolucijom 320x320 piksela. U platformu su još uključeni kamera, modul za slušalice sa mikrofonom, modul sa kontrolnim dugmićima i senzori vezani za određivanje pozicije i kretanja [19].

U procesu rada na portovanju smo koristili javno dostupan U-Boot i Linux kernel za Newton2 [20]. Kernel koji smo preuzeli dolazio je u dve varijante. Prva je bila namenjena za standardne Linux distribucije, a druga za Android verziju sistema. Najveći broj *driver*-a je funkcionisao bez krupnih izmena, a ALSA i frame buffer *driver*-i su zahtevali veći napor za ostvarivanje specifikovanih funkcionalnosti. ALSA *driver* je u potpunosti portovan, što je odrađeno po uzoru na Dorado ploču sa M200 *SOC*-om. Portovanje frame buffer *driver*-a je ostvareno uz značajnu podršku inženjera iz kompanije Ingenic Semiconductor.

U prateći softver koji dolazi uz Newton2 ulazi i aplikacija nazvana Cloner koja služi za snimanje osnovnog sadržaja operativnog sistema na fizički uređaj. Putanje ka prevedenim binarnim fajlovima za U-Boot i Linux kernel su se direktno dodavale u Cloner, kao i putanja za EXT4 Tizen sliku sistema. Pomenuta slika sistema je kreirana pravljenjem prazne EXT4 slike sistema željene veličine u koju je kopiran sadržaj .tar Tizen slike sistema kreirane pomoću MIC alata.

Početna pretpostavka je bila da je ranije opisani problem sa opcijom *--gc-sections* izazivao greške samo prilikom prevođenja i da promenljive u kojima smo ga dodatno viđali nisu uticale na linkovanje koda koji smo preveli. Ovo se u završnici pokazalo kao naš najveći propust, s obzirom da je dovelo do problema prilikom izvršavanja koda. U fazi početnog pokretanja, sistem je posle nekoliko minuta prestajao sa odzivom i prelazio u takozvano *hang* stanje. Analiza problema je utvrdila da su pretpostavke bile pogrešne i da je opcija *--gc-sections* za koju je smatrano da nema uticaja na generisani kod dovodila do generisanja velikog broja *dump core* fajlova, koji su ukazivali na *segmentation fault* i *reserved instruction* greške. Nakon realizacije da postoji propust, ceo kod je detaljno pregledan i uticaj opcije *--gc-sections* je neutralisan.

Jedan od bitnih problema sa kojima smo se susreli bila je greška prilikom pokušaja instalacije aplikacija. Akcija instalacije bi započinjala, ali bi svaki put bila prekidana sa sledećom greškom:

```
processing result : FATAL_ERROR [61] failed
```

Analiza problema je utvrdila da je potrebno iskoristiti *dbus IPC* umesto *default sockets IPC*, što je omogućeno sa sledeće dve izmene:

```
diff --git a/CMakeLists.txt b/CMakeLists.txt
index 8e6b396..c66d85e 100644
--- a/CMakeLists.txt
+++ b/CMakeLists.txt
@@ -71,8 +71,8 @@ ADD_DEFINITIONS("-std=c++0x")            # No
warnings about deprecated featu

# DBUS_CONNECTION and ENABLE_PRIVACY_MANAGER should be
enabled together to make
# dbus and privacy manager code enabled
-ADD_DEFINITIONS("-DSOCKET_CONNECTION")       #defines sockets
as used IPC
-#ADD_DEFINITIONS("-DDBUS_CONNECTION")        #defines DBus as
used IPC
+#ADD_DEFINITIONS("-DSOCKET_CONNECTION")      #defines
sockets as used IPC
+ADD_DEFINITIONS("-DDBUS_CONNECTION")        #defines DBus as
used IPC
 IF(ENABLE_PRIVACY_MANAGER EQUAL 1)
 ADD_DEFINITIONS("-DENABLE_PRIVACY_MANAGER")  #disables
privacy manager from wrt-secuirty-daemon
 ENDIF()
```

```
diff --git a/packaging/wrt-security.spec b/packaging/wrt-security.spec
index 3020e08..3ebb441 100644
--- a/packaging/wrt-security.spec
+++ b/packaging/wrt-security.spec
@@ -63,6 +63,9 @@ export LDFLAGS+="-Wl,--rpath=%{_libdir}"
 %cmake . -DDPL_LOG="ON" \
       -DVERSION=%{version} \
       -DCMAKE_BUILD_TYPE=%{?build_type:%build_type} \
+%ifarch mipsel
+       -DDBUS_CONNECTION=1 \
+%endif
 %if 0%{?enable_wrt_ocsp}
       -DENABLE_WRT_OCSP=1 \
 %else
```

IV. VERIFIKACIJA FINALNOG REŠENJA

Tizen sadrži dve grupe automatizovanih testova za verifikaciju: Web TCT i Native TCT. Native TCT se sastoji iz sledeće 3 podgrupe: CTC, ITC i UTC. Rezultati testova izvršenih na Newton2 su prikazani u tabeli I. Kao referentni sistem je korišćen Tizen emulator za Intel arhitekturu, čiji rezultati su prikazani u tabeli II.

TABELA I
REZULTATI VERIFIKACIONIH TESTOVA NA NEWTON2 PLATFORMI

Testovi	Prošlo	Palo	Blokirano	Neizvršeno
Web TCT	4469	321	369	1417
CTC	64	25	0	0
ITC	216	793	0	0
UTC	1485	1764	2	602

TABELA II
REZULTATI VERIFIKACIONIH TESTOVA NA INTEL EMULATORU

Testovi	Prošlo	Palo	Blokirano	Neizvršeno
Web TCT	7762	166	4	0
CTC	73	12	4	0
ITC	624	339	34	284
UTC	2237	583	141	0

Direktno poređenje rezultata nije jednostavno, pošto se u zavisnosti od hardverskih komponenti određuje koji testovi će da se pokrenu, što rezultuje mogućim variranjem broja testova na različitim platformama. Iz priloženoga je vidljivo da Tizen na Newton2 zaostaje za Intel emulatorom. Dobijeni rezultati nam nisu predstavljali prepreku, pošto je zadati fokus bio na manuelnim testovima, a rešavanje delikatnih problema je odloženo za neku od sledećih iteracija.

Manuelni testovi poput reprodukcije zvuka, video reprodukcije, Bluetooth povezivanja, pokretanja instaliranih aplikacija i opšte navigacije kroz grafički korisnički interfejs su funkcionisali bez detektovanih problema.

V. PREZENTACIJE FINALNOG REŠENJA

Newton2 sa Tizen 2.3 wearable operativnim sistemom je predstavljen na IoT Developers Conference 2016 od strane Imagination Technologies.

Za potrebe konferencije preuzet je kod aplikacija dostupan u primerima aplikacija iz Tizen SDK i preveden je bez modifikacija. Ideja je bila da napravimo sliku sistema pomoću MIC alata, na nju dodamo aplikacije i omogućimo drugim korisnicima da je samo presnime na uređaj posle preuzimanja te da u prvom podizanju sistema sve bude funkcionalno. Pošto aplikacije treba da se instaliraju dok je sistem u radu, nismo mogli na jednostavan način da ih dodamo u gotovu sliku sistema. Problem je rešen tako što smo sve prevedene aplikacije dodali u direktorijum */opt/usr/apps/demo* i kreirali servis koji će osiguravati da se aplikacije instaliraju pozivajući skriptu za instalaciju. Da bi se skripta izvršila u odgovarajuće vreme, kada su podignuti svi potrebni servisi, u */usr/lib/systemd/system/multi-user.target.wants/* direktorijumu smo napravili simbolički link *na demo-install.service*, koji ima sledeći sadržaj:

```
[Unit]
Description=Installing demo applications

[Service]
Type=simple
ExecStart=/usr/bin/demo_install.sh
```

Zatim smo kreirali */usr/bin/demo-install.sh* skriptu koja će da odradi potrebne instalacije:

```
#!/bin/sh

WGT_PKG_PATH="/opt/usr/apps/demo"
WGT_INSTALL_CMD="pkgcmd -i -t wgt -p"
LIST_INSTALLED_APPS_CMD="pkgcmd -l -t wgt"
DISPLAY_TIMEOUT_CMD="vconftool set -f -t int
db/setting/lcd_backlight_normal 300"

installed_apps="$($LIST_INSTALLED_APPS_CMD)"

for wgt_pkg in $( ls $WGT_PKG_PATH ); do
   if [[ "$wgt_pkg" != *".wgt"* ]]; then
      continue
   fi
   echo "Searching for $wgt_pkg ..."
   wgt_pkg_name=${wgt_pkg%.wgt}
   if [[ "$installed_apps" != *"$wgt_pkg_name"* ]]; then
      #Set display timeout to 300s to avoid need for interaction during the first
boot
      $DISPLAY_TIMEOUT_CMD
      echo "not installed"
      echo "Installing $wgt_pkg ...";
      $WGT_INSTALL_CMD "$WGT_PKG_PATH/$wgt_pkg"
   else
      echo "installed"
   fi
done
```

Nakon pokretanja sistema aplikacije će biti instalirane i prikazane kao na slici 1.

Finalno rešenje sa mehanizmom automatske instalacije aplikacija je takođe predstavljeno i na Tizen Experts portalu, najvećoj svetskoj *web* stranici sa informacijama i vestima vezanim za Tizen. U izveštaju su navedene opšte informacije i priložen je video na kome se vidi sistem prilikom rada [21].

Slika 1. Tizen 2.3 wearable operativni sistem na Newton2 platformi

VI. Zaključak

Postavljeni zahtevi su uspešno ostvareni i podrška za MIPS32 arhitektu je implementirana u okviru operativnog sistema Tizen. Kreiran je inicijalni skup paketa koji omogućava jednostavniji početak rada na portovanju drugih Tizen profila ili novijih Tizen wearable verzija sistema. Veliki broj kreiranih programskih izmena je u sličnom obliku primenljiv na pakete iz drugih Tizen profila. Dobijen je funkcionalan Tizen operativni sistem na fizičkom uređaju. Pored uloge da služi kao *proof of concept*, obezbeđuje i platformu za nastavak rada na portovanju i implementaciji dodatnih funkcionalnosti. Takođe, može da se koristi i kao referentni sistem.

Zahvalnica

Zahvaljujemo se Danielu Kneževiću, Milanu Jelisavčiću i Oliveru Petroviću, koji su bili sastavni deo našeg tima za portovanje. Takođe se zahvaljujemo inženjerima iz kompanije Ingenic Semiconductor na nesebičnoj pomoći oko problema sa Linux kernel *driver*-ima. Zahvaljujemo se i menadžmentu kompanije Imagination Technologies na iniciranju projekta portovanja i obezbeđivanju potrebne logistike.

Ovaj rad je delimično finansiran od strane Ministarstva za prosvetu, nauku i tehnološki razvoj Republike Srbije, na projektu broj: III_044009_1

Literatura

[1] About Tizen, tizen.org/about, *Tizen Project.*
[2] J. S. Matulac, "Case Study of Tizen Operating System", University of Philippines Open University, Laguna, Philippines, 2016.
[3] MIPS32 Architecture, imgtec.com/mips/architectures/mips32/, *Imagination Technologies.*
[4] MIPS Processors, imgtec.com/mips/, *Imagination Technologies*
[5] K. Yaghmour, "Embedded Android: Porting, Extending, and Customizing", *O'Reilly Media*, 2013.
[6] MIPS Android, community.imgtec.com/developers/mips/android/, *Imagination Technologies.*
[7] M. Jelisavčić, N. Veljković, L. Tršić, D. Šićarov, "*Prilagođavanje operativnog sistema Firefox OS za arhitekturu MIPS32*", TELFOR, 2015.
[8] MIPS Linux, community.imgtec.com/developers/mips/linux/, *Imagination Technologies.*
[9] Zhangjin, W., Mc Guire, N. "*Porting RT-preempt to Loongson2F*". In Eleventh Real-Time Linux Workshop (p. 169).
[10] Developer Tools, community.imgtec.com/developers/mips/tools/, *Imagination Technologies.*
[11] Git Build System, source.tizen.org/documentation/reference/git-build-system, *Tizen Project.*
[12] MIC Image Creator, source.tizen.org/documentation/reference/mic-image-creator, *Tizen Project.*
[13] Getting Started Guide, source.tizen.org/documentation/developer-guide/getting-started-guide, *Tizen Project.*
[14] Toolchains, elinux.org/Toolchains, *Embedded Linux Wiki.*
[15] Architecture emulation containers with binfmt_misc, lwn.net/Articles/679308/, *LWN.*
[16] Fixed binfmt mask for mipsel, review.tizen.org/gerrit/#/c/30285/, *Tizen Gerrit Code Review.*
[17] Wearable target kickstart file, download.tizen.org/snapshots/2.3-wearable/common/tizen-2.3-wearable_20160904.1/builddata/images/target/image-configurations/wearable_target.ks, *Tizen Project.*
[18] Wearable emulator kickstart file, download.tizen.org/snapshots/2.3-wearable/common/tizen-2.3-wearable_20160904.1/builddata/images/emulator/image-configurations/wearable_emulator.ks, *Tizen Project.*
[19] Newton2 Features, ingenic.com/en/?newton/id/1/lm/1.html, *Ingenic Semiconductor.*
[20] Newton2 Getting Started, ingenic.com/en/?newton/id/1/lm/2.html, *Ingenic Semiconductor.*
[21] Video Demo, Tizen Ported onto MIPS based Newton2 Platform, tizenexperts.com/2016/08/video-tizen-2-3-wearable-mips-newton2-demo/, *Tizen Experts.*

Abstract

This paper describes the process of porting Tizen 2.3 wearable operating system for 32-bit MIPS processor architecture. Tizen is an operating system intended for a broad spectrum of devices, with an idea to integrate all supported devices into a unique ecosystem with a consistent user experience. MIPS is a RISC processor architecture that achieves decent performance under low power consumption, with a particularly dominant position in the domain of network devices. This publication describes the operating system and hardware platform in more details, exposes the reasons that caused porting to take place, describes different phases of work and shows verification test results.

Porting Tizen operating system for the MIPS32 architecture

Dragan Čečavac, Dejan Latinović and Petar Jovanović