4η εργαστηριακή άσκηση

Σχεδιασμός Ενσωματωμένων Συστημάτων

Αρβανίτης Χρήστος: 03114622
Μπαγάκης Μάνος: 03114157

Άσκηση 1

Χτίσιμο του cross compiler

Κατά το χτίσιμο του cross compiler, παρατηρήθηκε ένα σφάλμα το οποίο δεν μπορέσαμε να το επιλύσουμε προσθαφαιρώντας κάποιο από τα πακέτα. Επίσης, κοιτάζοντας το αρχείο build.log, παρατηρήσαμε το ακόλουθο σφάλμα:

Error while building gcc-6.3.0/gcc/ubsan.c:1474:23: error: ISO C++ forbids comparison between pointer and integer [-fpermissive]

Οι οδηγίες που παρέχονται για την άσκηση συνιστούν την εγκατάσταση του crosstool-ng από το repository που φιλοξενείται στο git://crosstool-ng.org/crosstool-ng. Παρόλα αυτα, η νεότερη έκδοση, φιλοξενείται σε ένα Github repository με το ακόλουθο url:
https://github.com/crosstool-ng/crosstool-ng
Στη τελευταία έκδοση του master branch το προηγούμενο bug έχει διορθωθεί οπότε και χρησιμοποιήσαμε αυτή την έκδοση. Όλα τα βήματα πλην του αρχικού ακολουθήθηκαν κατά γράμμα οπότε και επιτύχαμε το build του cross compiler.

Ερώτημα 1

Χρησιμοποιούμε την αρχιτεκτονική arm-cortexa9_neon-linux-gnueabihf διότι αυτή ακριβώς υποστηρίζει το QEMU target machine της πρώτης άσκησης. Αυτό γίνεται σαφες αν εκτελέσουμε εντός του qemu την εντολή uname -a οπότε και θα δούμε ακριβώς την ίδια αρχιτεκτονική με αυτή που χρησιμοποιήσαμε για τον cross compiler μας.

Ερώτημα 2

Σχετικά με τη βιβλιοθήκη C που χρησιμοποιήθηκε, αυτή είναι η glibc.

Ερώτημα 3

Εκτελούμε τη παρακάτω εντολή

~/x-tools/arm-cortexa9_neon-linux-gnueabihf/bin/arm-cortexa9_neon-linux-gnueabihf-gcc -O0 -Wall -o phods_crosstool.out phods_init.c

Στη συνέχεια εκτελούμε τις ζητούμενες εντολές:
file phods_crosstool.out
phods_crosstool.out: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 3.2.0, with debug_info, not stripped

Η εντολή αυτή μας παρέχει εντολές για το είδος του παραγόμενου αρχείου. Όπως βλέπουμε, ανάμεσα στις πληροφορίες που παρέχονται, αναγράφεται ότι προορίζεται για την ARM αρχιτεκτονική και ότι είναι ELF (δηλαδή εκτελέσιμο) για 32-bit αρχιτεκτονικές.

readelf -h -A phods_crosstool.out
ELF Header:
  Magic: 7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class: ELF32
  Data: 2's complement, little endian
  Version: 1 (current)
  OS/ABI: UNIX - System V
  ABI Version: 0
  Type: EXEC (Executable file)
  Machine: ARM
  Version: 0x1
  Entry point address: 0x1045c
  Start of program headers: 52 (bytes into file)
  Start of section headers: 14724 (bytes into file)
  Flags: 0x5000400, Version5 EABI, hard-float ABI
  Size of this header: 52 (bytes)
  Size of program headers: 32 (bytes)
  Number of program headers: 9
  Size of section headers: 40 (bytes)
  Number of section headers: 37
  Section header string table index: 36
Attribute Section: aeabi
File Attributes
  Tag_CPU_name: "7-A"
  Tag_CPU_arch: v7
  Tag_CPU_arch_profile: Application

```
  Tag_ARM_ISA_use: Yes
  Tag_THUMB_ISA_use: Thumb-2
  Tag_FP_arch: VFPv3
  Tag_Advanced_SIMD_arch: NEONv1
  Tag_ABI_PCS_wchar_t: 4
  Tag_ABI_FP_rounding: Needed
  Tag_ABI_FP_denormal: Needed
  Tag_ABI_FP_exceptions: Needed
  Tag_ABI_FP_number_model: IEEE 754
  Tag_ABI_align_needed: 8-byte
  Tag_ABI_align_preserved: 8-byte, except leaf SP
  Tag_ABI_enum_size: int
  Tag_ABI_VFP_args: VFP registers
  Tag_CPU_unaligned_access: v6
  Tag_MPextension_use: Allowed
  Tag_Virtualization_use: TrustZone
```

Χρησιμοποιώντας την readelf, λαμβάνουμε αναλυτικότερες πληροφορίες για το εκτελέσιμο που μόλις κάναμε compile.

Το μέγεθος του εκτελέσιμου ανέρχεται στα 16204 Bytes

Ερώτημα 4

Με compile απο τον linaro παράγεται εκτελέσιμο 13020 Bytes.

Τα δύο εκτελέσιμα έχουν παραχθεί με διαφορετικές βιβλιοθήκες C με την glibc να είναι μια γενικής φύσεως βιβλιοθήκη ενώ η newlib να είναι embedded systems oriented.

Επιπλέον, εκτελώντας την readelf για το νέο εκτελέσιμο, συγκρίνουμε με αυτό της προηγούμενης άσκησης. Το μεγαλύτερο σε μέγεθος, που παράχθηκε με το crosstool υποστηρίζει VFPv3[-D32] αρχιτεκτονική, οπότε και επιβάλλει 32 bit μεγέθους καταχωρητές, σε αντίθεση με το ελαφρύτερο VFPv3-D16 που επιβάλλει 16 bit καταχωρητές, με οποιαδήποτε εντολή προσπαθεί να έχει πρόσβαση σε 32 bit καταχωρητές να επιστρέφει UNDEFINED.

Ερώτημα 5

Αν και χρησιμοποιούν διαφορετικό library, τα δύο αυτά (newlib και gclib) υλοποιούν το ίδιο ISO C API οπότε αφού το interface δεν αλλάζει εκτελούνται σωστά στο target μηχάνημα.

Ερώτημα 6

Το αρχείο που παράγεται με το linaro έχει μέγεθος 4084552 bytes ενώ αυτό από το crosstool έχει μέγεθος 4173344 bytes. Παρατηρούμε μία διαφορά στο μέγεθος των εκτελέσιμων,, η οποία οφείλεται στη διαφορετική βιβλιοθήκη που χρησιμοποιεί το κάθε ένα εκτελέσιμο.

Ερώτημα 7

Α. Στο host μηχάνημα θα εκτελεστεί κανονικά, αφού οι αλλαγές υπάρχουν στον glibc του host και έγινε με αυτή τη library το χτίσιμο τόσο του cross compiler όσο και του ίδιου του my_foo.c

Β. Προφανώς στο target μηχάνημα δεν υπάρχει η ανανεωμένη υλοποίηση του glibc οπότε και το εκτελέσιμο δεν θα λειτουργήσει σωστά. Συγκεκριμένα δεν θα μπορέσει να εκτελέσει την my_foo συναρτηση.

C. Σε αυτή τη περίπτωση, το link δεν γίνεται δυναμικά κατά το runtime αλλά στατικά οπότε ο απαραίτητος κώδικας της βιβλιοθήκης συνοδεύει αυτόν του εκτελέσιμου. Αναμένουμε να λειτουργήσει φυσιολογικά το πρόγραμμα.

Άσκηση 2

Χτίσιμο του πυρήνα

Κατά το χτίσιμο του πυρήνα, αντιμετωπίσαμε το πρόβλημα μη συμβατότητας του συστήματός μας με το Linaro, καθώς τα binary executables του είναι 32 bit έκδοσης και το σύστημά μας 64bit αρχιτεκτονικής. Προκειμένου να αντιμετωπίσουμε το πρόβλημα αυτό, και δεδομένου πως χρησιμοποιούμε Debian based distribution, εκτελέσαμε τα ακόλουθα:

```
sudo dpkg --add-architecture i386

sudo apt-get update
sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386
```

Επιπλέον, αν και ακολουθήσαμε τα βήματα του οδηγού, η έκδοση debian του guest μηχανήματος διαθέτει τέτοιο dpkg εκτελέσιμο, ώστε να μην υποστηρίζεται extract με συμπίεση tar.xz. Συγκεκριμένα εμφανίστηκε το ακόλουθο σφάλμα κατά την εγκατάσταση των πακέτων:

```
dpkg-deb: file `*.deb` contains ununderstood data member data.tar.xz     , giving up
```

Προκειμένου να λύσουμε αυτό το πρόβλημα χωρίς να αναβαθμίσουμε κάποιο πακέτο στο guest μηχάνημα, αποσυμπιέσαμε τα deb στο Host και τα επανασυμπιέσαμε χρησιμοποιώντας την συμβατή tar.gz συμπίεση. Συγκεκριμένα ακολουθήσαμε τα εξής βήματα:

dpkg-deb -R package.deb tmp
rm package.deb
sudo dpkg-deb -Zgzip -b tmp package.deb
rm -rf tmp
Τέλος ξανά-ανεβάσαμε τα αρχεία .deb στο guest μηχάνημα.
Στη συνέχεια, όλα λειτούργησαν όπως ήταν αναμενόμενο βάσει των οδηγιών.

Ερώτημα 1

Στον νέο πυρήνα εκτελούμε uname -a με τα ακόλουθα αποτελέσματα:
Linux debian-armhf 3.2.102 #1 SMP Wed Jan 2 21:09:39 EET 2019 armv7l GNU/Linux

Συνεπώς η έκδοση πυρήνα μας είναι η 3.2.102

Ερώτημα 2

Προκειμένου να προσθέσουμε μια κλήση συστήματος στον υπάρχοντα πυρήνα, ακολουθούμε τα εξής βήματα:

Προσθέτουμε στο αρχείο arch/arm/kernel/calls.S, δηλαδή στον πίνακα κλήσεων συστήματος, την ακόλουθη γραμμή:

```
CALL(sys_chrisman)
```

Όπου sys_chrisman το όνομα της κλήσης συστήματος που θα υλοποιήσουμε.

Στο αρχείο arch/arm/include/asm/unistd.h προσθέτουμε την γραμμή

```
#define __NR_chrisman         (__NR_SYSCALL_BASE+ 7)
```

Αλλάξαμε αυτή τη γραμμή επειδή είναι η πρώτη κενή που βρήκαμε. Το 7 αποτελεί το offset της, δηλαδή το αναγνωριστικό της για το system call API.

Στο αρχείο arch/arm/kernel/sys_arm.c ορίζουμε την υλοποίησή μας:

```
asmlinkage long sys_chrisman()
{
	printk("Greetings from kernel and team 11\n");
	return 0;
}
```

Τέλος, επεξεργαζόμαστε το αρχείο /include/linux/syscalls.h όπου και προσθέτουμε το πρωτότυπο της υλοποίησής μας:

```
asmlinkage long sys_chrisman(void);
```

Είμαστε έτοιμοι να χτίσουμε τον νέο πυρήνα.

Ερώτημα 3

Το πρόγραμμα που χρησιμοποιήσαμε είναι το εξής:

```
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>

#define __NR_chrisman 7

long chrisman_syscall(void)
{
    return syscall(__NR_chrisman);
}

int main(int argc, char *argv[])
{
    long int a = chrisman_syscall();
    printf("System call returned %ld\n", a);
    printf("Check dmesg for more fun and the printk output\n", a);
    return 0;
}
```