ΣΧΕΔΙΑΣΜΟΣ ΕΝΣΩΜΑΤΩΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ, 2017-2018

Άσκηση 3: Cross-compiling προγραμμάτων για ARM αρχιτεκτονική

Εγκατάσταση cross-compiler building toolchain crosstool-ng

Κατά την διάρκεια εκτέλεσης των βημάτων, ενδέχεται κάποια απαραίτητα πακέτα να λείπουν από το σύστημά μας. Στην περίπτωση αυτή εγκαθιστούμε το αντίστοιχο πακέτο. Π.χ. Για το βήμα 1, χρειαζόμαστε το subversioning tool git. Αν δεν το έχουμε εκτελούμε sudo apt-get intall git. (Σε περίπτωση που έχουμε άλλο package manager, προσαρμόζουμε την εντολή σε αυτόν.)

Βήματα:

1. Εκτελούμε την εντολή
   git clone git://crosstool-ng.org/crosstool-ng

   Η εντολή θα δημιουργήσει στο directory που την εκτελέσαμε έναν φάκελο με όνομα crosstool-ng.
2. Μπαίνουμε στο φάκελο, και αρχικά εκτελούμε
   ./bootstrap
3. Στη συνέχεια, θα πρέπει να δημιουργήσουμε δύο φακέλους στο HOME directory μας ως εξής:
   mkdir $HOME/crosstool
   mkdir $HOME/src

   Στον πρώτο φάκελο θα εγκατασταθεί το πρόγραμμα crosstool-ng ενώ στον δεύτερο, θα αποθηκεύει τα απαραίτητα πακέτα που κατεβάζει για να χτίσει τον cross-compiler.
4. Εκτελούμε την παρακάτω εντολή για να κάνουμε configure την εγκατάσταση του crosstool-ng
   ./configure --prefix=${HOME}/crosstool
   Κατά τη διάρκεια της εκτέλεσης αυτής της εντολής θα εμφανιστούν πολλά πακέτα που λείπουν. Θα πρέπει να τα εγκαταστήσετε και στη συνέχεια να ξαναεκτελέσετε την παραπάνω εντολή. Ενδέχεται μερικά πακέτα να μην έχουν ίδιο όνομα στον package manager, όπως για παράδειγμα αν το configure βρει ότι λείπει το πρόγραμμα awk να πρέπει να εγκαταστήσετε το πακέτο gawk. Επιπλέον πακέτα που μπορεί να χρειαστούν είναι τα εξής: texinfo, libtool, libncurses5-dev, g++.
5. Εκτελούμε την εντολή make
6. Εκτελούμε την εντολή make install
7. Μέχρι εδώ, αν όλα πάνε καλά, πρέπει να έχει εγκατασταθεί το crosstool-ng. Πηγαίνουμε στο install path $HOME/crosstool/bin. Σε αυτό το φάκελο θα κάνουμε build τον cross compiler μας. Αν θέλουμε να εκτελέσουμε το build από άλλο φάκελο πρέπει να κάνουμε export το παραπάνω directory στο path μας.
8. Εκτελούμε την εντολή
   ./ct-ng list-samples
   Θα εμφανιστεί μία λίστα με πολλούς συνδυασμούς αρχιτεκτονικών, λειτουργικών συστημάτων και βιβλιοθηκών της C που παρέχονται από το εργαλείο για να μπορούμε να κάνουμε γρήγορα και σωστά configure το build του cross-compiler που θέλουμε να παράξουμε για συγκεκριμένο target machine. Εμείς θα επιλέξουμε την: arm-cortexa9_neon-linux-gnueabihf.

9. Εκτελούμε την εντολή
   ./ct-ng arm-cortexa9_neon-linux-gnueabihf για να παραμετροποιήσουμε το crosstool-ng
10. Εκτελούμε την εντολή ./ct-ng menuconfig, αν θέλουμε να αλλάξουμε με γραφικό τρόπο κάποια χαρακτηριστικά του preconfigured συνδυασμού target machine, όπως για παράδειγμα ποια βιβλιοθήκη της C θα χρησιμοποιήσετε.
11. Εκτελέστε την εντολή ./ct-ng build που θα αρχίσει να χτίζει τον cross compiler. Το χτίσιμο του cross compiler είναι μία σχετικά χρονοβόρα διαδικασία.
12. Αν όλα έχουν πάει καλά, θα έχει δημιουργηθεί ο φάκελος $HOME/x-tools/arm-cortexa9_neon-linux-gnueabihf όπου μέσα στον υποφάκελο bin περιέχει τα εκτελέσιμα αρχεία του cross compiler σας.

Παρατηρήσεις:

Στο φάκελο $HOME/src το crosstool-ng θα κατεβάσει και θα αποθηκεύσει τα απαραίτητα αρχεία για να χτίσει τον cross compiler του. Σε περίπτωση που αυτά δεν υπάρχουν θα τα κατεβάσει εκ νέου.

Σε περίπτωση που το crosstool-ng αδυνατεί να κατεβάσει κάποιο από τα αρχεία που χρειάζονται μπορείτε να τα κατεβάσετε εσείς manually και να τα τοποθετήσετε στον φάκελο $HOME/src και το crosstool θα τα βρει όταν ξεκινήσετε το επόμενο build.

Η διαδικασία του building περιλαμβάνει έναν αριθμό από steps. Μπορείτε να δείτε αυτά τα βήματα με την εντολή ./ct-ng list-steps.

Υπάρχει περίπτωση κάποιο από αυτά τα steps να μην εκτελείται σωστά και το building να σταματάει. Αν ένα βήμα από αυτά μπορεί να παραληφθεί για το χτίσιμο του compiler όπως πχ να μην έχετε debugging support, είστε ελεύθεροι να το κάνετε. Υπάρχει επίσης η δυνατότητα να συνεχίσετε το building process από συγκεκριμένο step. Περισσότερες πληροφορίες μπορείτε να βρείτε στο αρχείο crosstool-ng/docs/4 - Building the toolchain.txt.

Θα χρησιμοποιήσουμε και έναν pre-compiled cross compiler που παρέχεται από την ιστοσελίδα www.linaro.org.

1. Κατεβάζουμε τα binaries του cross compiler από την παρακάτω διεύθυνση: https://releases.linaro.org/archive/14.04/components/toolchain/binaries/gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux.tar.bz2
2. Παρατηρούμε στο κάτω μέρος της σελίδας ότι τα binaries του cross compiler συμπεριλαμβάνουν τα παρακάτω στοιχεία:
   - Linaro GCC 4.8 2014.04
   - Linaro Newlib 2.1 2014.02
   - Linaro Binutils 2.24.0 2014.04
   - Linaro GDB 7.6.1 2013.10
   - A statically linked gdbserver
   - A system root
   - Manuals under share/doc/
   - The system root contains the basic header files and libraries to link your programs against.
3. Τα binaries του cross compiler βρίσκονται στο πακέτο που κατεβάσαμε στον φάκελο /bin

Άσκηση 1

1. Γιατί χρησιμοποιήσαμε την αρχιτεκτονική arm-cortexa9_neon-linux-gnueabihf; Τι μπορεί να συνέβαινε αν χρησιμοποιούσαμε κάποια άλλη αρχιτεκτονική από το list-samples όταν θα τρέχαμε ένα cross compiled εκτελέσιμο στον QEMU και γιατί;
2. Ποια βιβλιοθήκη της C χρησιμοποιήσατε στο βήμα 10 και γιατί; (*Χρήσιμη εντολή: ldd)*
3. Χρησιμοποιώντας τον cross compiler που παρήχθει από τον crosstool-ng κάντε compile τον κώδικα phods.c με flags -O0 -Wall -o phods_crosstool.out από το 2ο ερώτημα της 1ης άσκησης (τον απλό κώδικα phods μαζί με την συνάρτηση gettimeofday()). Τρέξτε στο τοπικό μηχάνημα τις εντολές:
   file phods_crosstool.out
   readelf -h -A phods_crosstool.out
   Τι πληροφορίες μας δίνουν οι εντολές αυτές;
4. Χρησιμοποιώντας τον cross compiler που κατεβάσατε από το site της linaro κάντε compile τον ίδιο κώδικα με το ερώτημα 3. Βλέπετε διαφορά στο μέγεθος των δύο παραγόμενων εκτελέσιμων; Αν ναι, γιατί;
5. Γιατί το πρόγραμμα του ερωτήματος 4 εκτελείται σωστά στο target μηχάνημα εφόσον κάνει χρήση διαφορετικής βιβλιοθήκης της C;
6. Εκτελέστε τα ερωτήματα 3 και 4 με επιπλέον flag -static. Το flag που προσθέσαμε ζητάει από τον εκάστοτε compiler να κάνει στατικό linking της αντίστοιχης βιβλιοθήκης της C του κάθε compiler. Συγκρίνετε τώρα τα μεγέθη των δύο αρχείων. Παρατηρείτε διαφορά στο μέγεθος; Αν ναι, που οφείλεται;
7. Έστω ότι προσθέτουμε μία δική μας συνάρτηση mlab_foo() στη glibc και δημιουργούμε έναν cross-compiler με τον crosstool-ng που κάνει χρήση της ανανεωμένης glibc. Δημιουργούμε ένα αρχείο my_foo.c στο οποίο κάνουμε χρήση της νέας συνάρτησης που δημιουργήσαμε και το κάνουμε cross compile με flags -Wall -O0 -o my_foo.out
   A. Τι θα συμβεί αν εκτελέσουμε το my_foo.out στο host μηχάνημα;
   B. Τι θα συμβεί αν εκτελέσουμε το my_foo.out στο target μηχάνημα;
   C. Προσθέτουμε το flag -static και κάνουμε compile ξανά το αρχείο my_foo.c. Τι θα συμβεί τώρα αν εκτελέσουμε το my_foo.out στο target μηχάνημα;

Παραδοτέο της άσκησης είναι περιγραφή του πως ξεπεράσατε τα προβλήματα που ανέκυψαν κατά το χτίσιμο του cross-compiler και σύντομες απαντήσεις στις παραπάνω ερωτήσεις.

Άσκηση 2

Στα πλαίσια αυτής της άσκησης θα χτίσουμε έναν νέο πυρήνα για το Debian OS που τρέχαμε στις ασκήσεις 1 και 2. Τα απαραίτητα συστατικά για την επίτευξη αυτού του στόχου είναι:

1. Χρήση ενός cross compiler που χρησιμοποιήσαμε στο ερώτημα 1. Μπορείτε να χρησιμοποιήσετε όποιον από τους δύο θέλετε.
2. Ελέγξτε το directory /boot/ στο debian του qemu που περιέχει τα υπάρχοντα αρχεία του linux image και του initrd, για να μπορείτε να τα ξεχωρίσετε και να τα συγκρίνετε με αυτά που θα δημιουργηθούν στο τέλος της διαδικασίας εγκατάστασης νέου πυρήνα.
3. Χρειαζόμαστε τον source κώδικα του πυρήνα που θέλουμε. Θα μπορούσαμε να κατεβάσουμε έναν από το www.kernel.org ωστόσο η διαδικασία του να βρούμε τα κατάλληλα patches που χρειάζεται για να τρέξει στον QEMU είναι απαγορευτική. Ως εκ τούτου θα κατεβάσουμε τον πηγαίο κώδικα που μας παρέχει κατευθείαν το debian. Επομένως, μπαίνουμε στο guest μηχάνημα μας και εκτελούμε τις εντολές:

1. apt-get update
2. apt-get install linux-source

   Η εκτέλεση αυτή θα κατεβάσει στο directory /usr/src το αρχείο linux-source-3.2.tar.bz2

4. Το compiling του πυρήνα θα γίνει στο host μηχάνημα γιατί ο χρόνος που χρειάζεται για να γίνει στο guest μηχάνημα είναι απαγορευτικός. Αντιγράφουμε λοιπόν στο host μηχάνημα το αρχείο που κατεβάσαμε και το κάνουμε extract.
5. Για το configuration του πυρήνα για το target μηχανήμα θα χρησιμοποήσουμε πάλι ένα configuration αρχείο που παρέχει το debian. Κατεβάστε από το mycourses το αρχείο kernel_config_new και αντιγράψτε το στο untarred directory του linux-source με όνομα .config .
6. Για να κάνουμε configure τον πυρήνα με το συγκεκριμένο configuration αρχείο πρέπει να εκτελέσουμε την εντολή make oldconfig στο directory του kernel source. Ωστόσο, επειδή κάνουμε cross-compiling πρέπει να υποδείξουμε στο make την target αρχιτεκτονική και τον cross compiler που θα χρησιμοποιήσουμε. Ο ευκολότερος τρόπος είναι να εκτελέσουμε την εντολή ως εξής:

   sudo make ARCH=arm CROSS_COMPILE=<path_to_your_cross_compiler>/bin/arm-cortexa9_neon-linux-gnueabihf- oldconfig

   Παρατηρήστε ότι δε βάζουμε όλο το όνομα του cross compiler παρά μόνο το γενικό prefix κάθε αρχείου που έχει παραχθεί από το building process του cross compiler. Αυτό συμβαίνει διότι κατά τη διάρκεια της μεταγλώττισης του πυρήνα χρειάζονται διάφορα εργαλεία πέρα από τον gcc και για τον λόγο αυτό, παρέχουμε το γενικό prefix και αφήνουμε τη διαδικασία του make να επιλέξει ποια εργαλεία χρειάζονται.
7. Κατεβάστε από το mycourses το αρχείο builddeb_hf.patch και με αυτό κάντε patch το αρχείο scripts/package/builddeb.
8. Κανονικά τώρα θα εκτελούσαμε μία εντολής της μορφής του βήματος 6, απλά αντί για oldconfig θα είχαμε all για να μεταγλωττίσουμε τον πυρήνα μας. Το αποτέλεσμα της διαδικασίας θα ήταν ένα compressed image του πυρήνα στον φάκελο arch/arm/boot με το όνομα vmImage. Η εγκατάσταση του πυρήνα στο debian, ακολουθεί μια πιο συστηματική διαδικασία, για να εξασφαλίσει την σωστή εγκατάσταση του πυρήνα στο target μηχάνημα. Πιο συγκεκριμένα, θα δώσουμε οδηγία στο make να δημιουργήσει 3 *.deb αρχεία που θα περιέχουν:

   i. Το καινούριο image του πυρήνα
   ii. Τα ανανεωμένα kernel headers
   iii. Ένα ακόμα πακέτο με headers από τον πυρήνα του Linux που χρησιμοποιούνται για userspace προγραμματισμό, μέσω την libc και των βιβλιοθηκών συστήματος.

   Για να συμβούν τα παραπάνω πρέπει να δώσουμε στο make το directive deb-pkg. Τέλος με το flag -j και έναν αριθμό μεγαλύτερο του 1 μπορείτε να εκτελέσετε το make σε παραπάνω από έναν επεξεργαστές για να επιταχυνθεί η διαδικασία. Συνολικά η ελάχιστη μορφή της εντολής είναι:

sudo make ARCH=arm CROSS_COMPILE=<path_to_your_cross_compiler>/bin/arm-cortexa9_neon-linux-gnueabihf- deb-pkg

   Με το πέρας της εντολής, θα έχουν δημιουργηθεί 3 deb αρχεία στο parent directory του linux source. Με την εντολή dpkg –info file_name.deb μπορείτε να πάρετε περισσότερες πληροφορίες για το κάθε ένα από αυτά τα αρχεία.
9. Ανεβάστε τα αρχεία στο target μηχάνημα και εγκαταστήστε τα με dpkg –i packet_name.deb

10. Η εγκατάσταση του linux-image πακέτου, θα εγκαταστήσει στο /boot/ directory του target συστήματος σας το νέο image του πυρήνα και ένα ανανεωμένο initrd. Για να λειτουργήσει ορθά το σύστημά σας με τον νέο πυρήνα, πρέπει να δίνετε το image και το initrd σαν ορίσματα στον qemu όταν τον τρέχετε. Επομένως, κατεβάστε τα αρχεία αυτά στο host μηχάνημα και επανεκκινήστε τον qemu δίνοντας τα ως ορίσματα στα flags kernel και initrd.

Ερωτήματα:

1. Εκτελέστε uname -a στον qemu εφόσον έχετε εγκαταστήσει τον νέο πυρήνα και σημειώστε το όνομα του νέου σας πυρήνα.
2. Προσθέστε στον πυρήνα του linux ένα καινούριο system call που θα χρησιμοποιεί την συνάρτηση printk για να εκτυπώνει στο log του πυρήνα την φράση "Greeting from kernel and team no %d" μαζί με όνομα της ομάδας σας.
3. Γράψτε ένα πρόγραμμα σε γλώσσα C το οποίο θα κάνει χρήση του system call που προσθέσατε.

Παραδοτέα θα είναι:

I. Τα αρχεία του source του πυρήνα που πραγματοποιήσατε αλλαγές
II. Το τελικό image του πυρήνα με το system call σας
III. Το πρόγραμμα του ερωτήματος 3
IV. Μια σύντομη αναφορά με την απάντηση στο ερώτημα 1, τυχόν προβλήματα που αντιμετωπίσατε στο χτίσιμο του πυρήνα και τα βήματα που χρειάζονται για να γίνει σωστά η μεταγλώττιση του πυρήνα με το system call σας.

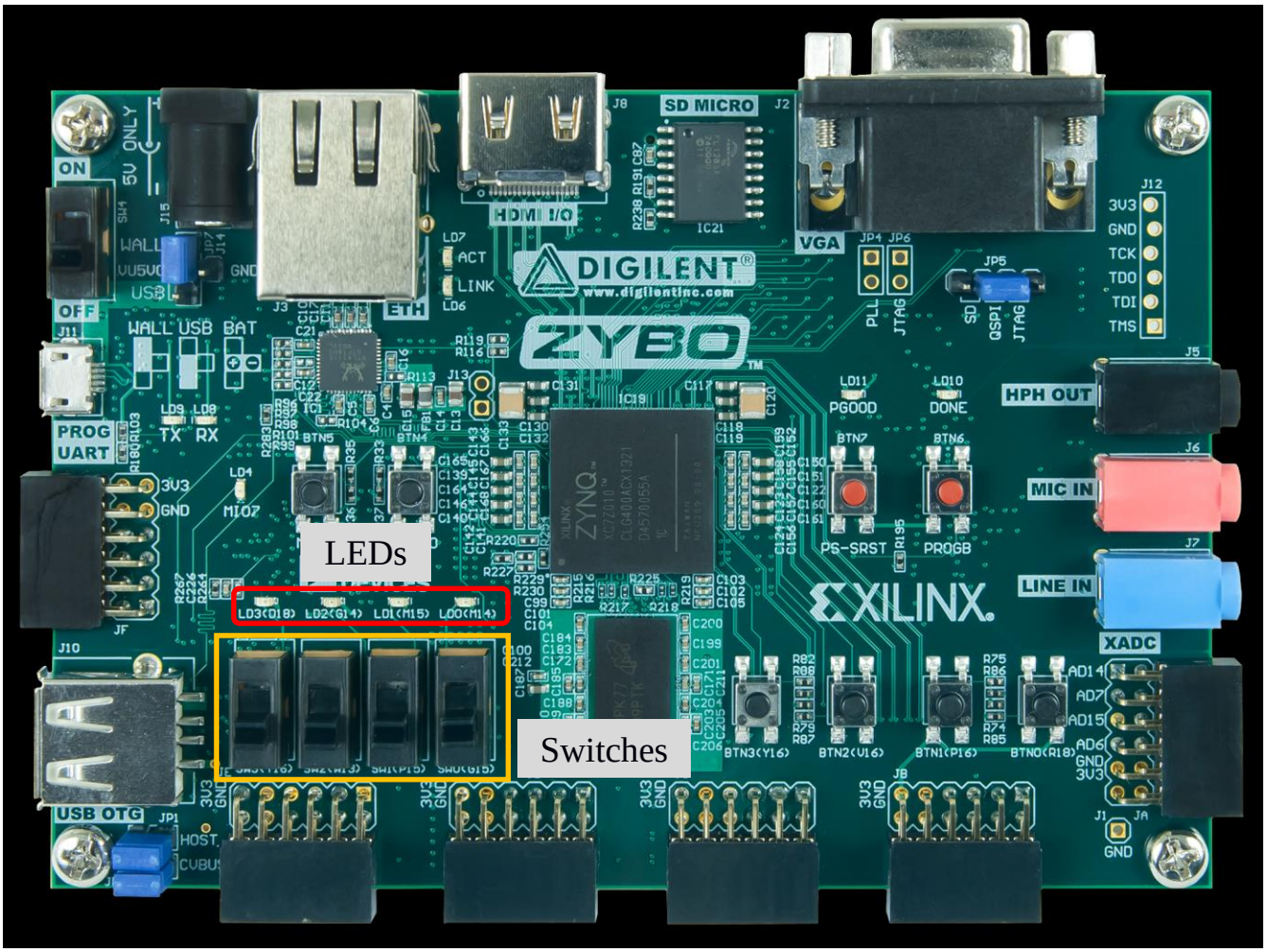
Bonus Άσκηση (Προαιρετική) – 1 μονάδα

Η αναπτυξιακή πλατφόρμα Zybo Zynq-7000 (https://www.xilinx.com/products/boards-and-kits/1-4azfte.html), περιέχει ένα ενσωματωμένο σύστημα σε ψηφίδα (SoC), το οποίο περιλαμβάνει δύο επεξεργαστές ARM Cortex™-A9 και FPGA fabric. Το σύστημα υποστηρίζει την εκτέλεση λειτουργικού συστήματος Linux, καθώς και Freertos v9.0. Η δυνατότητα προγραμματισμού του συστήματος με Freertos, παρέχεται ενσωματωμένη στο Xilinx Software Development Kit, παρέχοντας έτσι την δυνατότητα να γίνει χρήση των βιβλιοθηκών της Xilinx.

Στην Εικόνα 1 φαίνεται η αναπτυξιακή πλατφόρμα Zybo. Επί του σχήματος φαίνονται δύο παραλληλόγραμμα, τα οποία αντιστοιχούν στα 4 LEDs καθώς και τους 4 διακόπτες (DIP switches) της πλατφόρμας. Σκοπός της άσκησης είναι να γραφτεί ένα πρόγραμμα με βάση το Freertos το οποίο θα κάνει χρήση αυτών των περιφερειακών. Κάθε LED θα αναβοσβήσει με μια συγκεκριμένη περίοδο, η οποία θα εξαρτάται από την θέση του αντίστοιχου dip switch. Συγκεκριμένα η σχέση εισόδου των switches και περιόδου ON/OFF του αντίστοιχου LED είναι:

Dip Switch 0	**LED 0 blinking period**
Input = 0	500 ms
Input = 1	1000 ms
Dip Switch 1	**LED 1 blinking period**
Input = 0	1000 ms
Input = 1	2000 ms
Dip Switch 2	**LED 2 blinking period**
Input = 0	1500 ms
Input = 1	3000 ms

Dip Switch 3	LED 3 blinking period
Input = 0	2000 ms
Input = 1	4000 ms



Εικόνα 1:Digilent Zybo Zynq-7000

Η λειτουργία του κάθε LED να αντιστοιχίζεται σε ένα διαφορετικό task του Freertos. Το διάβασμα των τιμών των dip switches να πραγματοποιείται με polling της τιμής και όχι με interrupt σε περίπτωση μεταβολής. Σημαντικό κομμάτι της άσκησης είναι να γίνει σωστά η διαμόρφωση του FPGA fabric, ώστε να συνδεθεί η είσοδος/έξοδος των επεξεργαστών με το GPIO της πλατφόρμας. Η διαμόρφωση αυτή θα πραγματοποιηθεί με αυτοματοποιημένο τρόπο, μέσω του Xilinx Vivado. Παραδοτέο της άσκησης είναι ο πηγαίος κώδικας, μια σύντομη αναφορά των βημάτων ανάπτυξης καθώς και επίδειξη της ορθής λειτουργίας στο εργαστήριο Μικροϋπολογιστών.

Για περισσότερες πληροφορίες και διευκρινίσεις απευθυνθείτε στο vtsoutsouras@central.ntua.gr.