Programmation assembleur (ASM)

Magali Frölich / Daniel Rossier

Environnements de développement croisé

28 septembre 2014
Echéance: 13 octobre 2014, 23h59

Objectif du laboratoire

L'objectif de ce laboratoire est de se familiariser avec les outils de développement croisé, principalement dans un environnement émulé. De plus, nous nous familiariserons avec *U-boot*, notre principal moniteur dans le cadre ce cours. Différents exercices s'effectueront sur une plate-forme émulée type x86 et ARM.

Consignes de laboratoire

- Le laboratoire est évalué selon les conditions établies dans le document annexe qui vous a été distribué en début de semestre.

Etape no. 1 - *Patch* de la machine virtuelle

a) Pour ce laboratoire, nous allons travailler avec une plate-forme émulée type x86. Pour compiler votre code vers cette plateforme vous aurez besoin de la toolchain *i686-pc-linux-gnu*.

Une archive est disponible à cette adresse : ***http://heig.ch/demeshi***
La commande suivante permet d'extraire la toolchain dans le dossier *opt*.

```
$ sudo tar xf i686-pc-linux-gnu.tar.bz2 -C /opt    Copie la toolchain dans opt
```

b) Pour l'utiliser, il faut encore ajouter les lignes suivantes à la fin du fichier */home/redsuser/.bashrc* :

```
$ nano .bashrc                                     Edite le fichier .bashrc

PATH="$PATH":/opt/i686-pc-linux-gnu/bin/           Ligne à ajouter
export PATH                                        Ligne à ajouter
```

Rappel : Dans *nano*, on sauve les modifications avec ctrl+X.

Etape no. 2 - Mise à jour du *workspace asm_student*

a) Commencez par vous assurer que votre *workspace* est à jour. Pour cela faites un *commit* de vos modifications locales. A partir de votre *shell*, effectuez les commandes suivantes à la racine de *asm_student* :

```
$ git add fichier1 fichier2          spécifie les fichiers à sauver
$ git commit                         synchronise votre dépôt local
$ git fetch asm_student_orig         importe les commits effectués sur le serveur
$ git merge asm_student_orig/master  synchronise votre dépôt local avec le serveur
```

Indications

- Attention ! Comme le dépôt sur le serveur contient le répertoire *.metadata* nécessaire au fonctionnement d'*Eclipse*, et que ces métadonnées évoluent en fonction de l'utilisation d'*Eclipse*, vous pouvez préserver vos données localement en renommant provisoirement le répertoire *.metadata* avant un *git pull*, ou plus simplement vous pouvez effacer complètement le répertoire *.metadata* et le récupérer du dépôt avec la commande *git checkout .metadata*

```
$ mv .metadata tmp
$ git pull
$ rm -rf .metadata
$ mv tmp .metadata
(ou)
$ git checkout .metadata
```

Etape no. 3 - Premiers pas avec *U-boot* et déploiement d'applications

Cette étape est dédiée à l'apprentissage d'*U-boot*. Pour cela, nous allons utiliser le projet ***c_x86***.

Vous constaterez que, dans votre *workspace*, les projets se trouvent dans deux répertoires principaux, en fonction du langage de programmation: ***c*** et ***asm*.** A l'intérieur de ces deux sous-répertoires se trouvent deux autres sous-répertoires associés à la machine-cible: ***x86*** et ***arm***. Pour faciliter l'accès aux projets, des liens symboliques ont été créés à la racine du *workspace*: ***asm_arm, asm_x86, c_arm*** et ***c_x86***.

a) Dans un *shell*, allez dans le répertoire de projet ***c_x86*** et lancez la commande *make* afin de compiler l'application *hello_world*.
b) Depuis la racine du *workspace*, démarrez le script *./st86*.

⇒ Que contient ce script ? Expliquez les différentes options.

c) Sur la fenêtre principale (de laquelle vous avez lancé le script, vous pourrez observer les traces du démarrage du moniteur. Vous êtes maintenant dans le *shell* d'*U-boot*. Tapez la commande *help* et examinez les commandes suivantes: ***go, md, mm, mw, printenv, setenv, tftp***.

d) Démarrez l'application *hello_world* avec la commande suivante:

```
boot > run tftphello
boot > go 0xef00000
```

⇒ Quel est le sens de la première commande ?

Etape no. 4 - *Debugger* une application tournant dans l'émulateur *qemu*

a) Examinez dans *eclipse* la configuration de *debug* intitulée "*C - x86 - hello_world*" (examinez les options dans l'onglet *Commands*)
b) Démarrer le *debugger* après avoir lancer le script *./st86-debug*

⇒ Expliquez précisément le sens de cette démarche ?

c) Activez un *breakpoint* à l'entrée de la fonction *hello_world()* et lancez l'exécution de votre programme.

⇒ Quelle est l'adresse de la prochaine instruction à exécuter ?

d) Ouvrez le *Makefile* de cette application et identifiez la règle de *linkage*.

⇒ Quelles sont les options du *linker* pour cette application ?

Etape no. 5 - Exécution d'une application *cross-compilée* pour *ARM*

a) Lancez la commande *make* dans le projet ***c_arm***.

⇒ Quelle est la *toolchain* utilisée dans ce projet ?

b) Exécutez l'application *hello_world* selon le même principe que l'étape no.2, mais cette fois-ci en utilisant le script ***./stf***
c) Examinez le contenu de ce script

⇒ Quelles sont les différences avec la version *st86* ?

d) Tapez la commande *printenv* afin de visualiser les variables d'environnement dans cette version d'*U-boot*.

Dans cette version, vous pourrez sauvegarder vos variables d'environnement avec la commande *saveenv*. Les variables d'environnement seront sauvegardées dans une mémoire *flash* virtuelle.

e) Activez un *breakpoint* avant la fonction *getc()* et examinez le code désassemblé de *hello_world* (en utilisation la bonne configuration de *debug*). Utilisez pour ceci le script *./stf-debug*

⇒ En observant l'exécution dans le code source, tout en faisant du pas-à-pas sur le code désassemblé, que peut-on conclure sur l'implémentation de la fonction *getc()* ?

Etape no. 6 **- Déploiement d'un code assembleur *cross-compilé* pour *ARM***

Pour cette étape, il faut utiliser le projet ***asm_arm***.

a) Compiler le code assembleur du fichier *asmA1.S* en utilisant le *Makefile*.
b) Exécuter l'application en *debug* jusqu'à l'instruction "*mov fp, sp*" et observez le code désassemblé correspondant.

⇒ Quelle est l'adresse de cette instruction ?
⇒ Choisissez une instruction quelconque du programme en cours d'exécution et donnez-en sa représentation en hexadécimal.
⇒ Quelles constations pouvez-vous faire entre le code source et le code désassemblé ?