Programmation assembleur (ASM)

Magali Fröhlich / Daniel Rossier

Jeux d'instructions (première partie)

29 octobre 2015
Echéance: 17 novembre 2015, 23H59

Objectif du laboratoire

Ce laboratoire permet d'aborder les instructions de traitement en assembleur *x86* et *ARM*. Il s'agira également de se familiariser avec les cartes de référence rapide et d'examiner les différences entre un code assembleur original et le résultat d'une compilation.

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 - Opérations arithmétiques et logiques

a) En supposant que le registre ***edx*** soit utilisé pour stocker la valeur d'un registre de périphérique, réalisez le code suivant en assembleur *x86*. Pour cela utilisez le fichier ***asm_x86/asmA1.S***

```
void config_register(void) {
  int gpio_reg = 0x4600;

  /* On suppose que la variable gpio_reg contient la valeur du registre… */
  gpio_reg |= 0x2;
  gpio_reg &= ~0x6;
  gpio_reg ^= 0x8;
  gpio_reg = (gpio_reg & ~0x38) | (0x6 << 3);
}
```

b) Editez un fichier C contenant ce code, compilez le pour l'architecture cible *x86* 32 bits, désassemblez le code binaire en utilisant l'application *objdump* de la *toolchain* correspondante, et comparez le résultat. Le fichier à créer est ***c/x86/asmA1.c***

⇒ Vous devrez rendre les deux fichiers sur ***bitbucket***. Veillez à respecter la structure du dossier :

- votre code source assembleur **:** *asm_student/c/x86/asmA1.S*
- le code désassemblé avec vos commentaires :
  pour cela complétez le fichier asm_student/README.MD.

Indications

- Adaptez le *Makefile* existant afin d'inclure votre fichier C.
- Respectez l'ordre des opérations proposées !

c) A l'aide du *debugger*, examinez le code en exécution de ce programme. Examinez les instructions *x86* produites et expliquez comment l'opérateur ~ a été traduit par le compilateur.

d) Refaites toutes les étapes précédentes pour une version de code en assembleur *ARM,* et en utilisant le registre *r8* pour stocker la valeur du registre de périphérique.

Attention utilisez l'application *objdump* de la *toolchain* correspondante.

⇒ Comme pour le point précédent, vous devrez rendre deux fichiers sur bitbucket : **votre code source assembleur** et le **code désassemblé** avec vos commentaires.

Etape no. 2 **- Algorithme de division**

a) Dans le projet ***asm_x86***, créez et éditez le fichier ***asmA2.S*** et implémentez la division de deux nombres entiers (*a/b*) sans opérateur de multiplication et de division. Stockez le quotient et le reste dans deux registres différents. Effectuez quelques tests avec des grands nombres.

Indications

- *a* et *b* sont deux entiers positifs non nuls.
- Pour cette étape, vous aurez besoin de quelques instructions de saut. Aidez-vous des cartes de référence rapide afin de trouver les bonnes instructions.

b) Faites de même en assembleur *ARM* dans le projet ***asm_arm***.

⇒ Vous devrez rendre les deux fichiers sur *bitbucket*. Veillez à respecter la structure du dossier :

- asm_student/asm_x86/asmA2.S
- asm_student/asm_arm/asmA2.S