SCR.2.1 TP 18 ⊥ :

Programmation en langage d'assemblage pour l'architecture ARMv8

Décalages - Codage des instructions

https://developer.arm.com/documentation/ddi0602/2022-12/Base-Instructions

INPUT : Les nombres à tester sont placés dans la section `.data`. On peut utiliser `sed` et sa commande `s` pour en changer de valeurs. On retient leur adresse à l'aide d'étiquettes `int1`, `int2`, etc.

OUTPUT : On utilise `gdb` pour observer l'évolution des valeurs de registres.

I. Instructions de décalage.

On va écrire le programme `shift-experiment.s` avec les instructions `lsl`, `lsr` et `asr`.

1. Placer la constante 1 dans le registre `x0`.
2. Ecrire une instruction avec `lsl` qui permet de multiplier par 8 le contenu du registre `x0`.
3. Ecrire une instruction avec `lsr` qui permet de diviser par 2 le contenu du registre `x0`.
4. Quelle est la différence entre `lsr` et `asr` ? Ecrire une suite d'instructions qui permet d'illustrer la différence entre les deux.

II. Utilisation du décalage dans la deuxième opérande.

Un décalage peut être réalisé sur la deuxième opérande de certaines instructions comme `add` ou `sub`. On va écrire le programme `barrel-experiment.s` qui illustre le *barrel-shifter*.

1. Placer la constante 15 dans le registre `x0`.
2. Compléter la ligne d'instruction suivante de manière à obtenir dans le registre `x1` 17 fois la valeur initiale contenue dans le registre `x0` :

```
add x1,x0,x0,lsl ...
```

3. Quelle est la valeur obtenue dans `x1` par l'exécution des instructions suivantes ?

```
mov x2,#1
sub x1,xzr,x2,lsl #3
```

III. Codage des instructions.

Utiliser la commande `objdump -d` pour désassembler le programme exécutable obtenu pour `barrel-experiment.s`. Sur combien de bits sont codées les instructions ? En cherchant dans la documentation ARM (voir lien URL indiqué au début du TP), comparer le codage de l'instruction `add x1,x0,x0,lsl ...` avec le format attendu. Expliquer.