

TD 5 - PROGRAMMATION EN ASSEMBLEUR

Arithmétique, appels de procédure, récursivité.

Page web : <http://www-rocq.inria.fr/~acohen/teach/archi.html>

Exercice 5.1 - Arithmétique multi-précision

Sur 16 bits, la valeur d'un nombre entier positif est comprise entre 0 et $2^{16} - 1$. Pour manipuler de grandes valeurs entières, il est nécessaire de créer des routines reproduisant les opérations arithmétiques sur de grands nombres, codés sur plusieurs mots. Dans cet exercice, on implémentera ces routines et on pourra ensuite les illustrer sur le calcul de la factorielle d'un entier.

On utilise deux nouveaux programmes pour faciliter l'assemblage et l'exécution sur le processeur LC-2.

- Script shell `_cohen/bin/lc2asm`.

À utiliser pour appeler l'assembleur LC-2 (éviter de lancer directement l'assembleur).

Utilisation :

```
> lc2asm fichier.asm fichier
```

- Produit `fichier.obj` : code "objet" exécutable sur le simulateur "logiciel" du LC-2 (voir ci-dessous, à ne pas confondre les circuits *DigLog*).

- Produit `fichier.lst` : mise en parallèle des instructions, étiquettes, code machine binaire, code machine hexadécimal et adresses des instructions.

- Produit `fichier.hex` : utilisé pour charger les composants SRAM8K de *DigLog* (voir ci-dessous).

- Et autres extensions (`.bin`, `.log`, `.sym`) inutiles ici.

Attention aux messages d'erreur (pas toujours très explicites).

- Script shell `_cohen/bin/xlc2sim`.

Simulateur "logiciel" du LC-2, beaucoup plus rapide — et un peu plus convivial — que les circuits *DigLog*.

Utilisation :

```
> xlc2sim
```

Suivre les menus du simulateur.

Question 5.1.1

Écrire en assembleur une fonction (appelée par JSR) qui ajoute R0 à R1 (deux entiers positifs en binaire sur 16 bits), puis range la somme sur 16 bits dans R0 et range la retenue dans R1.

Réponse

Le code source (commenté) répondant à toutes les questions de l'exercice est donné dans le fichier *factorielle.asm*.

L'addition avec retenue est difficile à effectuer sur le LC-2 : il est impossible d'accéder directement à la retenue calculer par l'additionneur. La solution proposée consiste à tester le bit 15 des opérands, par l'intermédiaire de branchements conditionnels. Pour ces tests, R0 et R1 sont interprétés en complément à 2 :

- si R0 et R1 sont positifs, il ne peut y avoir de dépassement et la retenue est mise à 0 :
- si R0 et R1 sont négatifs, il y a forcément un dépassement et la retenue est mise à 1 :
- si les signes sont différents, il y a dépassement lorsque la somme de R0 et R1 est positive ; la retenue n'est donc mise à 1 qu'à cette condition.

Question 5.1.2

Écrire une fonction réalisant la multiplication (non signée) de R0 par R1 avec un résultat sur 32 bits, en rangeant les bits de poids faible du résultat dans R0 et les bits de poids fort dans R1.

Réponse

On peut proposer deux types de solutions.

- La plus simple consiste à ajouter R0 autant de fois que la valeur de R1, en utilisant la fonction de la question précédente ; le programme est très inefficace, mais cette solution reste réaliste dans le cadre de l'exercice (factorielle d'un nombre inférieur à 30, à priori).
- La solution proposée est une variante de l'algorithme de multiplication par accumulation et décalage : la somme partielle est calculée à partir des bits de poids fort, en testant le bit 15 du multiplieur R1 et en ajoutant R0 si besoin, puis en décalant à gauche la somme partielle (stockée sur deux registres) et le multiplieur R1. Cette solution a été retenue à cause de la difficulté de réaliser un décalage à droite sur le LC-2 (nécessaire pour une multiplication « traditionnelle » de droite à gauche).

Question 5.1.3

Un nombre entier positif N est représenté en *multi-précision* de la manière suivante : à une adresse donnée en mémoire, la représentation binaire de N sur $16m$ bits est codée sur m mots consécutifs, en commençant par les bits de poids fort (*big endian*). En pratique, on supposera que m est fixé à la valeur 16.

Écrire une fonction effectuant la multiplication du registre R0 (entier sur 16 bits) par un entier sur $16m$ bits, avec un résultat sur $16m$ bits. On supposera que R1 indique l'adresse du mot de poids fort de l'entier sur $16m$ bits, et que le résultat est rangé en commençant à l'adresse indiquée par R2 (m mots doivent être préalablement réservés à cette adresse).

Réponse

La multiplication se fait de droite à gauche, en partant de $R1+m-1$ et $R2+m-1$, en une seule traversée. À chaque étape, la fonction de multiplication de la question précédente est utilisée, et deux additions (dont une avec retenue) permettent d'accumuler les résultats.

Attention : lors de la dernière étape, la retenue de l'addition et le mot de poids fort de la multiplication ne doivent pas être conservés, sous peine de dépassement de capacité (à gauche) du résultat sur 16m bits.

Question 5.1.4

Écrire un programme utilisant une procédure récursive pour calculer — en multi-precision — la factorielle de l'entier sur 16 bits contenu dans une variable nommée argument ; le résultat sera rangé en mémoire dans le tableau de m mots nommé Resultat. Vérifier par exemple que $17! = x0001437E EEC D8000$ en hexadécimal (c'est-à-dire 355687428096000 en base 10).

Réponse

La procédure récursive est très naturelle : on sauvegarde R7 dans la pile (R6 doit être positionné au début d'une zone libre suffisamment vaste) à chaque appel et on appelle la multiplication en multi-precision de la question précédente. Attention : le résultat de la multiplication doit être recopié dans une zone temporaire avant chaque appel récursif, puisque cette multiplication ne s'effectue pas « en place ». Voir factorielle.asm.