1 Problématique

Les ordinateurs sont structurés selon l'architecture de von Neumann. L'idée novatrice est de stocker les instructions de programme dans la mémoire, au même titre que les données. Le microprocesseur peut effectuer des calculs. L'étape suivante est de pouvoir communiquer nos instructions à l'ordinateur.

Comment demander à la machine d'exécuter une série d'instructions ?

2 Un premier simulateur

On se propose de modéliser l'architecture de von Neumann "à la main". Par groupe de trois personnes se répartir les rôles :

- **Mémoire** : elle dispose de vingt cellules numérotées. Chaque cellule peut contenir une donnée ou une instruction. La première cellule contient la première instruction du programme.
- **Unité arithmétique et logique** : elle réalise les instructions qui lui sont demandées. Elle dispose de dix cellules numérotées (*les registres)* qui peuvent contenir des données. L'UAL ne peut effectuer d'opérations que sur les données de ses registres.
- **Unité de contrôle** : elle joue le rôle de métronome. Elle ordonne quelle étape est en cours de réalisation.

L'UAL peut réaliser les instructions suivantes :

- *Copier* dans le registre n une valeur,
- *Charger* dans le registre n la valeur de la cellule m de la mémoire,
- *Stocker* la valeur du registre n dans la cellule de mémoire m,
- *Additionner* le registre n avec une valeur (ou un registre).

Activité 1 : La cellule 16 de la mémoire contient déjà le nombre 230. Votre programme doit additionner le contenu de la cellule 16 avec le nombre 49, puis stocker le résultat dans la cellule 20 de la mémoire.

3 Découverte du simulateur

3.1 Premier programme

Chaque microprocesseur possède des caractéristiques propres (nombre de registres, instructions...). Cependant nous pouvons noter de nombreuses similarités. En effet, l'architecture de von Neumann reste la norme et les principes de communication avec la mémoire sont respectées.

en fonction marque, époque, évolution ; noms et nombres des registres sont différents : sur x86 : registres EAX, EBX, ECX.

Le programme sur la page web ci-après simule une architecture de von Neumann :

https://www.peterhigginson.co.uk/ARMlite/

1MB=1MiB=1mébiBytes (méga binaire) = 2^{20} Bytes = 1048576 Bytes (octets)
PC = Program Counter = va de 4 en 4 car les adresses de cellules mémoire vont de 4 en 4
LR = Link Register = holds the address to return to when a function call completes.

Christophe Viroulaud

SP = Stack Pointer = pointeur de pile = garde une trace de la pile d'appel.
1 mot mémoire = 32 bits = 8 chiffres hex
Il y a 1048756 octets = 1048756*8 bits
Un mot mémoire = 4 octets = 32 bits
1048756÷4 = nombre de mot mémoire = 262144 ; ils ne sont pas tous affichés à l'écran

Activité 2 : Recherche dans la documentation

1. De combien de registres dispose l'UAL ?
2. Que représente le registre *PC* ?
3. Quelle est la taille d'une cellule (ou *mot mémoire*) en bits ?

La documentation (en anglais) est située sur cette page :

https://peterhigginson.co.uk/ARMlite/Programming%20reference%20manual.pdf

Activité 3 : Utilisation du simulateur

1. Repérer visuellement les principaux éléments de l'architecture de von Neumann.
2. En bas à droite du simulteur, placer le menu déroulant sur *Decimal (signed)*.
3. Retrouver et noter la syntaxe des instructions de l'activité 1.
4. Quelle instruction permet de terminer un programme ?
5. Traduire alors le programme de l'activité 1 sur le simulateur.
6. Lancer le programme en mode pas à pas avec le bouton 1

FIGURE 1 – Pas à pas

3.2 Instructions supplémentaires

Charger des données en mémoire est une première étape mais il semble limité de s'arrêter là. Les instructions *.InputNum* et *.WriteString* utilisées dans le code 1 ci-après permettent de travailler avec les entrées/sorties dans la console.
Également comparer deux valeurs est une manipulation importante dans un programme.

Observer les *status bits*.

Activité 4 : Comparaisons de valeur

1. Dans la documentation trouver la signification et la syntaxe des instructions *CMP, BEQ, BNE*.
2. Charger le code 1 dans le simulateur :

```
1    LDR R0,.InputNum
2    MOV R1,#10
3    CMP R0,R1
4    BNE NOTEGAL
5    MOV R2,#GAGNE
```


Christophe Viroulaud

```
6       STR R2,.WriteString
7       B SUITE
8   NOTEGAL:
9       MOV R2,#PERDU
10      STR R2,.WriteString
11  SUITE:
12      HALT
13  GAGNE:.ASCIZ "Bien joué!"
14  PERDU: .ASCIZ "Perdu!"
```

Code 1 – Programme mystère

3. Que fait ce programme ?
4. Que se passe-t-il si on retire la ligne 7 ? Donner la signification de cette instruction.

4 Création d'un programme : calcul de moyenne

Afin d'alléger sa charge de travail, le professeur principal de la classe demande aux élèves de NSI de produire un programme pour calculer les moyennes trimestrielles des élèves. Nous considérons que les élèves ont huit notes pour le trimestre.

Activité 5 :

1. Décrire sur papier, l'algorithme à réaliser.
2. Traduire cet algorithme en code dans le simulateur.
3. L'exécuter.

créer une boucle
LSR pour diviser ; S'aider de la documentation et éventuellement du net pour voir la syntaxe de LSR
(LSR R0,R1,#3) = divise R1 par 2^3
enregistrer les notes en mémoire ?

Christophe Viroulaud