Architecture des ordinateurs 1

Romain Deleuze

17 septembre 2019

Table des matières

Chapitre 1

Concepts de base

Lecture 1: Instructions

Mardi 17 juin 8 :30

1.1 Instructions

1. Une série d'instructions arrivent au cœur de l'ordinateur. Traitées par le cpu pour avoir un résultat. Exemple :
   - 1. Instruction : addition
   - 2. Données : 1 et 2
   - 3. Résultat : 3

   Notion de flux continu
2. Explications exemple :
   - 1. Lire les nombres
   - 2. Modifier les nombres
   - 3. Écrire les nombres

   Conséquence sur le design des composants -> Accélérer les actions.

3. En détails :
   - 1. Lecture/Écriture -> nécessite une zone modifiable pour le stockage des données
   - 2. Modifier -> Nécessite une ALU pour modifier les données. Celle-ci faisant partie du CPU. Partie du CPU qui execute le calcul.
   - 3. Manipuler les données -> nécessite un chemin, les bus.
     - 1. Data Bus -> ensemble des fils qui traitent les données.
     - 2. Instruction bus -> ensemble des fils qui transportent les instructions. L'ensemble des fils -> bus.
   - 4. Résultat -> les données résultantes vont être renvoyées dans la zone mémoire. (écrire)

1.2 Registres

1. Exemple de réflexion architecturale : Les registres proches de l'ALU ont de meilleurs performances dans le traitement de données. -> Registres internes au CPU. Registre -> petit espace de stockage, rapide incorporé au CPU.

2. Exemple de l'addition : A+B=C
   - 1. Obtenir depuis les registres sources (A et B)
   - 2. Additionner les nombres.
   - 3. Placer le résultat dans le registre de destination (C) -> Écrase ce qui se trouvait dans le registre(0 et 1) -> les registres ne sont jamais vides.
3. **La mémoire RAM**
   - 1. Les registres sont petits -> contiennent peu de données.
   - 2. Données utiles -> déplacées vers les registres(accumulateur).
   - 3. Le reste -> dans **main memory** composée de mémoire de type **RAM(Random access memory) -> Accès RW(Lecture/Écriture)**, supprimée si plus de courant, besoin de **rafraichir la mémoire** pour garder les infos.).
   - **RAM statique** -> conserve la info sans rafraichissement.

   Schema : Main memory : Instructions + données(lire) CPU = ALU + registres (modifier) in memory bus Écrire : Résultat -> données in main memory
4. Exemple Addition : A+B=C
   - 1. Charger les deux opérandes (A et B) depuis la mémoire centrale (main memory) vers deux registres sources.
   - 2. Additionner :
     - a. Lire le contenu des registres A et B
     - b. Additionner les contenus des registres A et B
     - c. Écrire le résultat dans le registre C
5. **Le programme : code stream** Le code -> série d'instructions de commandes qui dépend du type de CPU(X86, ARM...) -> précise ce que la machine doit faire. -> exemple : Entrée reset.

   Exemple :
   - 1. Instruction **load** pour charger les nombres depuis la mémoire vers les registres
   - 2. Instruction **add** pour que le ALU réalise l'addition.
   - 3. Instruction **store** pour que l'ordinateur place le résultat en mémoire.
6. **Catégorie d'instructions**
   - 1. Arithmétique : add, sub, mul, div
   - 2. Accès mémoire : load, store

— 3. Branchement (jump)

— 4. Logique (conditions) : and, or, not

7. **Architecture basique et format d'instructions** Basé sur une machine fictive : **AR1** composé de :

— 1. 1 ALU

— 2. 4 registres : A, B, C, D

— 3. 256 cellules mémoires addressable de 0 à 255

(a) **Format des instructions :**

```
instruction source1, source2, destination
```

Exemple :

```
add A, B, C
#Passage dans le compilateur transforme les nbs binaires.
```

(b) **Format des instructions d'accès mémoire**

```
Instruction source, destination
#exemple:
load \#12, A
#exemple d'un programme:
load \#12, A
load \#13, B
add A, B, C
store C, \#14
#Les donnees en source ne changent pas.
```

(c) Limites de l'exemple : Comment connaitre le contenu des cellules #12 et #13.

8. **Valeurs immédiates et jeux sur les addresses**

(a) Utilisation d'une valeur à la place d'une adresse.

```
add, A, 2, A
# Ajouter la valeur 2 au contenu du registre A et ecrire
    le result dans A (en ecrasant ce qui s'y trouvait)
#Utilisation du symbole # pour aller chercher une adresse.
```

(b) 2ème programme (utilisation de D sans savoir sa valeur) :

```
load #D, A ;Contenu de #D = 12 dans A
load #13, B ;Contenu de la cellule 13 dans B
add A, B, C ;Add A+B -> result dans C
store C, #14 ;Stocker C dans cellule 14
```

(c) 3ème programme :

```
load #11, D ;Contenu de la cellule 11 dans D(pointeur)
load #D, A ;Contenu de la cellule D dans A (cellule 11)
load #13, B ;Contenu de la cellule 13 dans B
add A, B, C ;Add A+B -> result dans C
store C, #14 ;Stocker C dans cellule 14
```

9. Résumé :

— 1. Mettre une valeur dans un registre

— 2. Récupérer cette valeur en tant qu'adresse de cellule mémoire.

   - 3. Lire le contenu de cette cellule.
   - 4. Charger ce contenu dans un registre.
   - 5. La mémoire est divisée en segments(bits consécutifs) :
     - 1. Certains stockent des données.
     - 2. D'autres stockent du code.

10. **Adressage relatif = adresse de base + offset** Exemples :

```
load #(D+108), A
store B, #(D+108)
```

11. **Les adresses mémoires et les nombres entiers** : sont stockés dans les mêmes registres appelés : **general purpose registers(GPRs)**
    AR1 : A,B,C,D = GPRs.
12. **Autres registres :** Registre de **statut & registre de contrôle**.
    Schema + bus de contrôle, tout est addressable.