Architecture des ordinateurs

S. Zertal

Laboratoire LI-PaRAD, Université de Versailles

E-mail: Soraya.Zertal@uvsq.fr

JEU D'INSTRUCTIONS

Jeu d'instructions : Définition

Définition :
Une instruction = un ordre d'effectuer une tâche précise portant sur des données précises et de fournir un résultat.

Plusieurs paramètres : ordre , données , résultat

Jeu d'instructions : Composition

Une instruction est un enregistrement de plusieurs champs :

Jeu d'instructions : Composition

Une instruction est un enregistrement de plusieurs champs :

- **Un code opération** : type d'instruction (ordre).

Jeu d'instructions : Composition

Une instruction est un enregistrement de plusieurs champs :

- **Un code opération** : type d'instruction (ordre).
- Plusieurs autres informations indiquant où se trouvent **les opérandes** nécessaires à son exécution (données), qu'ils soient sources (en lecture) ou cible (en écriture) pour ranger le résultat.

Opcode	Opérandes sources et cible

Table : Format général d'une instruction

Jeu d'instructions : Modes d'adressage

La technique de localisation des opérandes est appelée **mode d'adressage** et il en existe plusieurs, dont :

Jeu d'instructions : Modes d'adressage

La technique de localisation des opérandes est appelée **mode d'adressage** et il en existe plusieurs, dont :

- **immédiat :** la valeur de l'opérande est disponible *immédiatement*.
  Mode ne nécessitant aucun accès mémoire.

Jeu d'instructions : Modes d'adressage

La technique de localisation des opérandes est appelée **mode d'adressage** et il en existe plusieurs, dont :

- **immédiat :** la valeur de l'opérande est disponible *immédiatement*.
  Mode ne nécessitant aucun accès mémoire.
- **direct :** l'adresse de l'opérande est disponible. On l'utilise pour accèder *directement* à sa valeur en mémoire.
  Mode nécessitant un seul accès mémoire.

- **indirect :** l'adresse de l'adresse de l'opérande est disponible.
  On l'utilise pour accèder au mot mémoire qui contient l'adresse du mot mémoire où est stockée la valeur de l'opérande.
  Mode nécessitant deux accès mémoire.

Jeu d'instructions : Modes d'adressage

- **indirect :** l'adresse de l'adresse de l'opérande est disponible. On l'utilise pour accèder au mot mémoire qui contient l'adresse du mot mémoire où est stockée la valeur de l'opérande.
  Mode nécessitant deux accès mémoire.
- **indexé :** l'adresse est composée d'une adresse de base et d'un déplacement à partir de cette adresse. Mode nécessitant une opération de calcul pour générer l'adresse absolue de l'opérande.
  Mode nécessitant un calcul et un seul accès mémoire.

Jeu d'instructions : CISC vs RISC

Un jeu d'instructions est l'ensemble d'instructions désignant toutes les opérations que peut exécuter une machine (processeur).

Les processeurs sont classés par rapport à leurs jeux d'instructions, qui peuvent être :

- **CISC** (Complex Instruction Set Computer), exemple: PentiumPRO, Intel Core i7, Phenom d'AMD,
- **RISC** (Reduced Instruction Set Computer), exemple: UltraSPARC, ARM
  Une particularité MISC (Minimal Instruction Set Computer : RISC poussé à l'extrême).

Jeu d'instructions : Propriétés

Complet : qui permet l'écriture d'un programme en langage machine capable d'évaluer n'importe qu'elle fonction calculable en utilisant un espace mémoire de taille raisonnable.

Efficace : qui permet l'exécution rapide des fonctions les plus fréquentes en utilisant peu d'instructions.

Régulier : qui contient pour chacune des instructions, un opcode et un mode d'adressage attendu (expl. s'il y a un décalage à gauche, le décalage à droite doit exister)

Jeu d'instructions - Classification

Classification :

Il existe 3 grandes classes de jeux d'instrucions :

1. Architecture à Pile.
2. Architecture à accumulateur.
3. Architecture à GPR (General purpose registers)

Architecture à Pile :

Les opérandes sont pris dans la pile et le résultat de l'ALU est placé automatiquement en sommet de pile.

En plus des opérations de calcul, on dispose de 2 instructions pour empiler (**push**) et dépiler (**pop**).

Jeu d'instructions - Classification

Architecture à Pile : Exemple: C = A+B

push A

push B

add

pop C

Architecture à accumulateur :
L'une des opérandes est l'accumulateur par défaut (acc = registre spécial).
Exemple: C = A+B
Load A
add B
Store C

Architecture à GPR (General purpose registers) :
Registres à usage général

L'architecture dispose d'un ensemble de registres à utiliser au besoin pour tout type de calcul.

Les opérandes explicites sont des registres ou des mots mémoire. Cette classe est celle la plus utilisée, pour la facilité de programmation et l'efficacité de la compilation.

Architectures à GPR :

1. Intérêt dû à l'usage effectif des registres par le compilateur : pour le calcul des expressions ou le stockage des variables.
2. Fléxibilité du calcul des expressions (comparé à la pile)
   Exemple : $(A \times B) - (C \times D) + (E/F)$
3. Possibilité de garder les variables dans les registres et réduire le traffic de/vers la mémoire.

Jeu d'instructions

Complexité des machines :

1. Complexité des formats de données
2. Complexité des formats d'instructions
3. Complexité du jeu d'instructions

Jeu d'instructions

Complexité des machines :

1. Complexité des formats de données
2. Complexité des formats d'instructions
3. Complexité du jeu d'instructions

Conséquences :

1. Complexité du contrôle
2. Complexité du compilateur
3. Instructions complexes $\Rightarrow$ Temps d'exécution long (temps de cycle important)

Jeu d'instructions : Optimisation

La vitesse d'exécution d'une instruction repose sur celle de son **décodage** : identification de son type et des opérandes.

Jeu d'instructions : Optimisation

La vitesse d'exécution d'une instruction repose sur celle de son **décodage** : identification de son type et des opérandes.

Des instructions simples de longueur fixe et un nombre réduit de champs réguliers $\Rightarrow$ temps de décodage réduit.

Jeu d'instructions : Optimisation

La vitesse d'exécution d'une instruction repose sur celle de son **décodage** : identification de son type et des opérandes.

Des instructions simples de longueur fixe et un nombre réduit de champs réguliers $\Rightarrow$ temps de décodage réduit.

Idée de base :
Faire en sorte que les instructions les plus **fréquentes** soient les plus **rapides** $\Rightarrow$ les plus simples.

Jeu d'instructions : Optimisation

Problème :
Les accès mémoire sont lents, donc pénalisants pour la vitesse d'exécution.

Solution : Restreindre les accès mémoire aux instructions LOAD et STORE.

Jeu d'instructions : Optimisation

Problème :
Les accès mémoire sont lents, donc pénalisants pour la vitesse d'exécution.

Solution : Restreindre les accès mémoire aux instructions LOAD et STORE.

Conséquence :

1. Toutes les instructions arith/log s'opèrent sur des registres.
2. Le recouvrement entre l'accès aux données et le calcul.

Processeurs RISC

Seul un nombre réduit d'instructions est utilisé $\Rightarrow$ Processeurs RISC (Reduced Instruction Set Computer) [David Patterson]

1. Peu d'instructions, peu de mode d'adressage, peu de format de données
2. Taille fixe des instructions
3. Opérandes en registres (accès rapide)
4. Contrôle cablé donc rapide

Processeurs RISC : Bilan

Les processeurs RISC sont rapides :

Processeurs RISC : Bilan

Les processeurs RISC sont rapides :

Instructions simples $\Rightarrow$ temps d'exécution court $\Rightarrow$ temps de cycle court.

Processeurs RISC : Bilan

Les processeurs RISC sont rapides :

Instructions simples $\Rightarrow$ temps d'exécution court $\Rightarrow$ temps de cycle court.

Conception rapide donc en phase avec la technologie.

Processeurs RISC : Bilan

Les processeurs RISC sont rapides :

Instructions simples $\Rightarrow$ temps d'exécution court $\Rightarrow$ temps de cycle court.

Conception rapide donc en phase avec la technologie.

Compilation rapide $\Rightarrow$ optimisations plus faciles à développer.