Architecture et Système

Stefan Schwoon

Cours L3, 2025/2026, ENS Paris-Saclay

Développement historique

Les premiers ordinateurs (années 50) :

peu d'instructions et signaux $\rightarrow$ architecture câblée

Les années 60-80 : âge de la microprogramation

jeux d'instructions et signaux toujours plus complexe :
microprogrammation plus facile à construire et gerer

une même architecture peut être adapté aux besoins différents

microprogrammation utilisateur sur certaines machines

Exemple: Jeu d'instructions Intel x86

Jeu d'instructions complexe, avec quelques instructions assez puissants (boucles pour traiter des blocs de mémoire)

Opérandes de 8/16/32 bits (pour compatibilité en arrière)

Instructions peuvent utiliser un registre partiel ou complet (AL/AH, AX, EAX)

Longueur d'instructions variable (1 à 7 octets), décodage compliqué

Pipeline / Chaîne de traitement

Si on sous-divise le calcul en plusieurs cycles, chaque instruction prend un nombre variable de cycles (IF, ID, EX1, EX2, . . .)

On sous-divise alors le circuit du processeur en plusieurs unités indépendantes qui s'occupent d'une tâche précise. Ainsi, l'exécution des instructions peut se chevaucher :

Au premier cycle, on exécute la phase IF de la première instruction.

Au second cycle, la phase ID de la première et la phase ID de la seconde.

etc...

Problème: Gestion des dépendances, branchements (conditionnels), longueur d'instructions et exécutions inégales, . . .

L'architecture RISC

A partir des annes 80 : retour au mode câblé

Facteurs technologiques :

miniaturisation rend possible des circuits plus complexes

outils pour automatiquement concevoir et arranger des circuits (CAD)
$\rightarrow$ construction des circuits complexes devient plus facile

apparition de l'architecture RISC qui permet des optimisations

RISC = reduced instruction set computing

Architecture RISC

Idée en général : uniformiser les instructions afin de les exécuter plus efficacement.

Caracteristiques typiques :

éliminer des opérations complexes, juste load/store/op.arithmétiques

mémoire rapide integrée dans le processeur (ou proche)

éliminer la phase *décodage* : codes opération toujours d'une même longeur, opérands toujours dans la même place de l'opcode.

exécution parallèle : exécuter plusieurs instructions à la fois : une instruction en phase "instruction fetch", une autre en "exécution"

plus de registres pour minimiser les transferts vers la mémoire

Inconveniences de RISC

Incompatible avec des architectures existantes (notamment x86)

Plus de travail pour les compilateurs

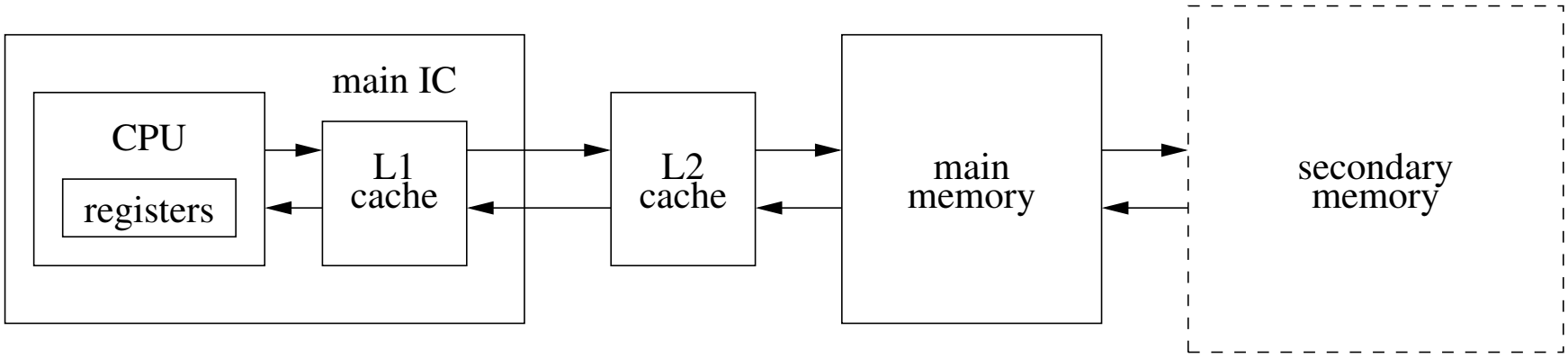
Mots d'instructions très grands, code machine peu compact

$\rightarrow$ combinaison des deux techniques:

(pré-)processeur traduit les opérations complexes vers (plusieurs) instructions RISC

une autre couche opère sur ces instructions RISC

Les caches



Idée: garder les données utilisées souvents dans une mémoire spéciale rapide, mais de taille limitée

Facteurs limitants : coût, espace physique limité

Cache à plusieurs niveaux, les cache distants sont plus lents mais aussi plus grands

Transparent : Le programmeur accède à une adresse virtuelle, le contenu se trouve dans la mémoire principale ou dans l'un des caches.

DRAM vs SRAM

Rappel: DRAM plus compact (un seul transistor + capaciteur)

lecture déstructive, fuite dans les capaciteurs $\rightarrow$ récharge périodique

compact mais lent, typiquement utilisé pour mémoire principale

SRAM (*static random-access memory*)

- réalisation par une bascule D
- moins compact mais plus rapide, utilisé pour les caches
- L1 = intégré dans le CPU, L2 = dans un autre chip

Une adresse dans le cache L1 est fournie trés rapidement ($\sim$ 4 cycles).

Pour récupérer une adresse dans la mémoire principale, il faut une centaine de cycles !

Parallelisme dans les architectures modernes

Le processeur exécute plusieurs instructions en parallèle, en profitant des différentes phases d'exécution (pipelining).

Plusieurs unités d'exécution travaillant en parallèle (superscalaire).

En principe, cela permet d'exécuter plus d'instructions dans une même temps. Mais il y a des problèmes :

- dépendances : le résultat d'une instruction est nécessaire pour le prochain
- branchements : quelle instruction sera la prochaine ?

Ces problèmes s'aggravent grâce aux différentes vitesses de mémoire.

Solutions :

- analyse des dépendences, exécution "out of order"
- exécution spéculative (sur un autre jeu de registres), prédiction des branchements