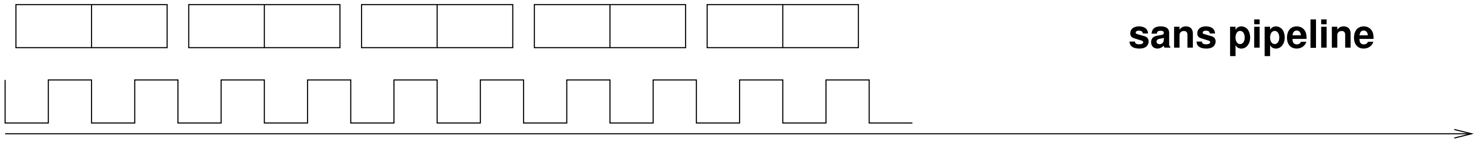
Pipeline

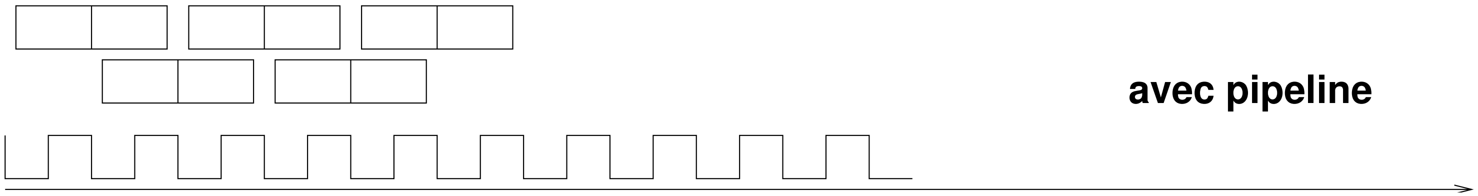
C’est une façon d’augmenter le nombre d’instructions exécutées par unité de temps sans diminuer le temps d’exécution des instructions

Il faut donc exécuter plusieurs instructions en parallèle

Exemple, dans notre architecture, les instructions arithmétiques n’utilisent pas les bus. On peut alors charger l’instruction suivante en même temps



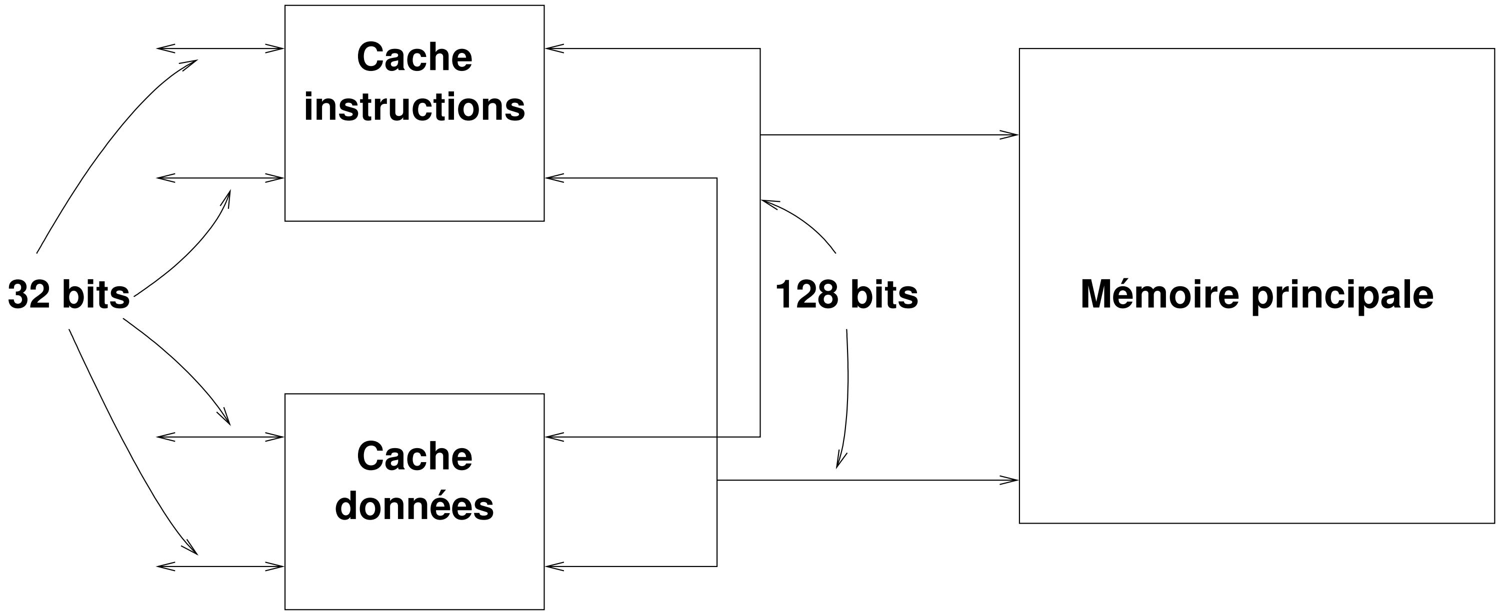
sans pipeline



avec pipeline

Pipeline

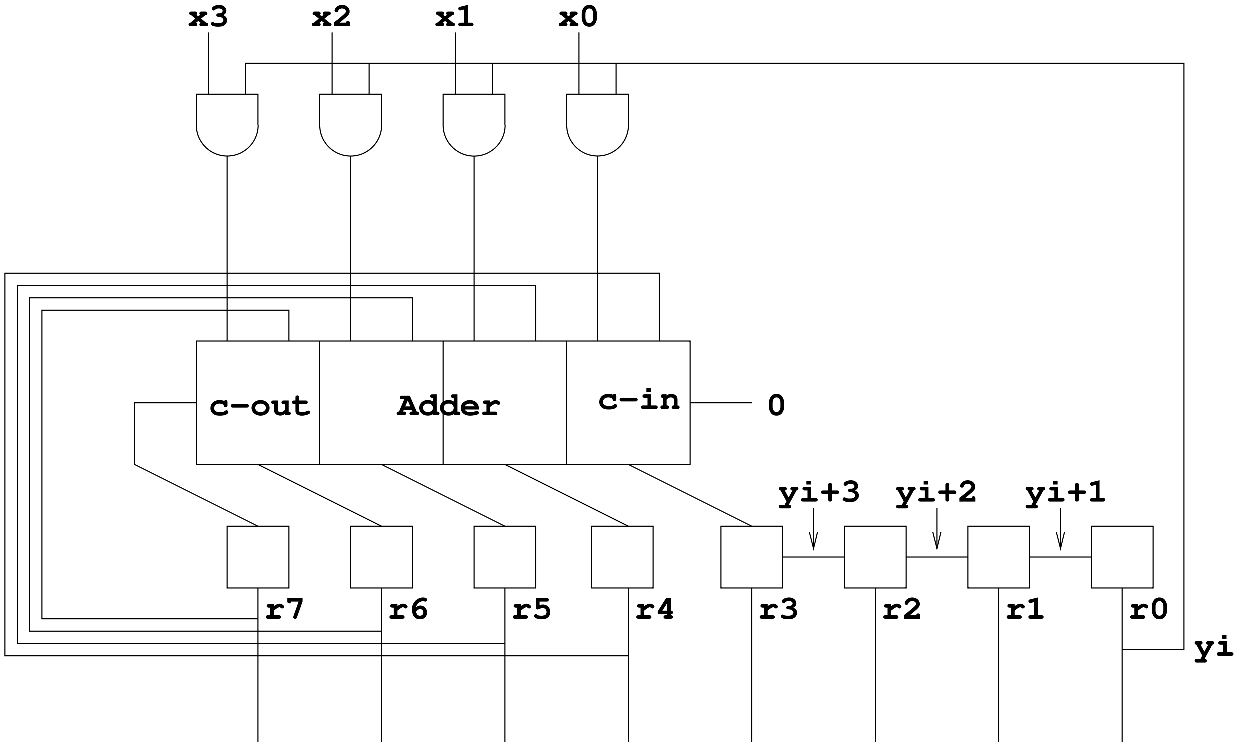
On peut souvent éviter l'utilisation simultanée du bus par le chargement et l'exécution d'une instruction même si son exécution a besoin de données en mémoire. Pour cela il faut un cache pour les instructions et un cache pour les données:



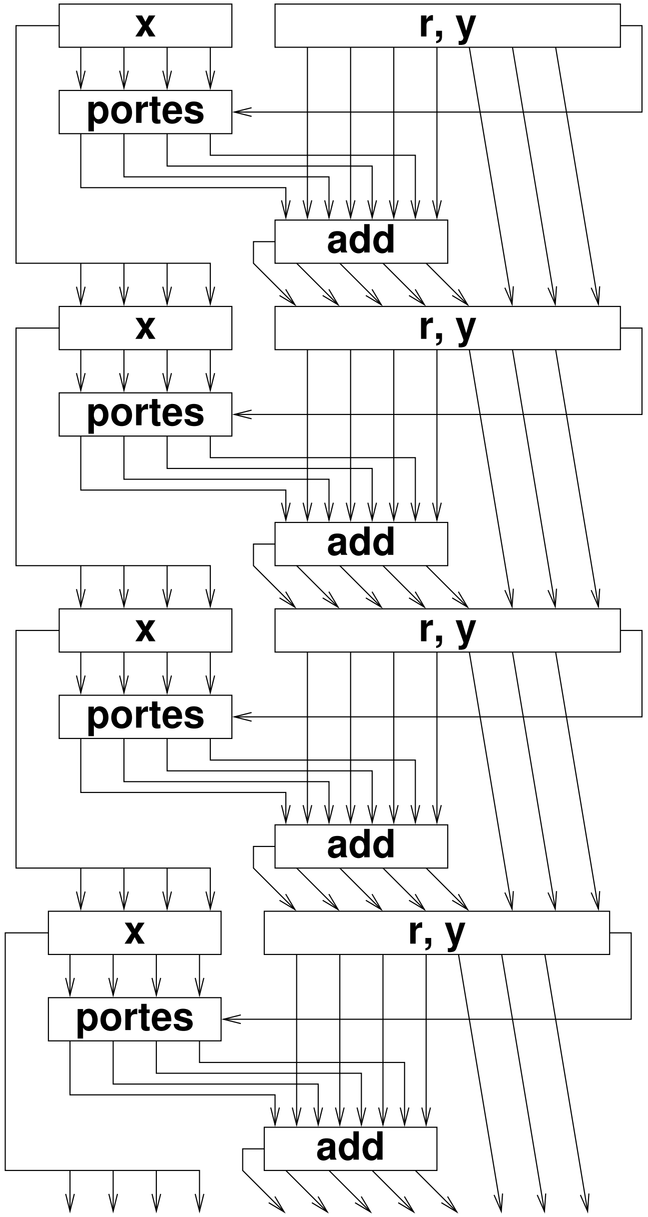
Pipeline

Pour exécuter plusieurs instructions simultanément, il faut souvent rajouter des circuits supplémentaires.

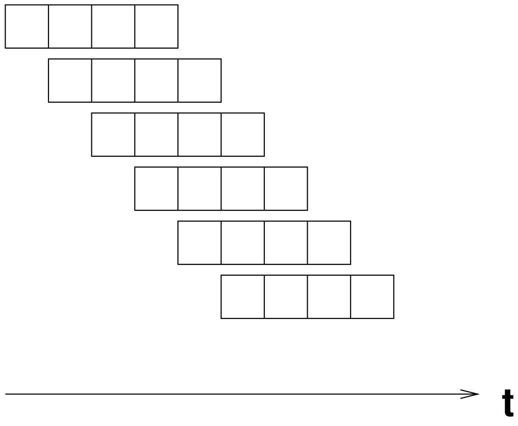
Exemple: multiplication



Pipeline



Exécution:



Multiprocesseurs

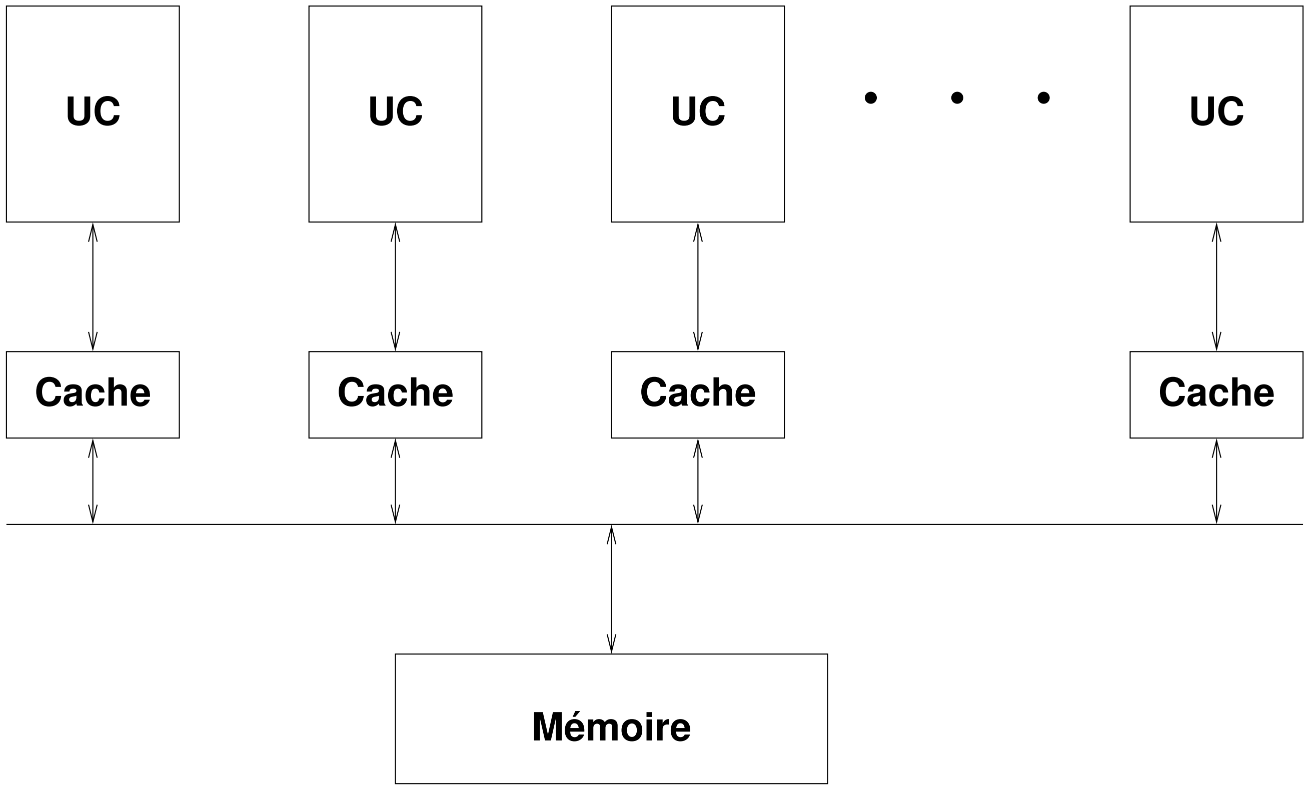
Plusieurs types selon si la mémoire est partagée ou distribuée et selon la méthode d'interconnexion.

Nous allons parler du type SMP (Symmetric Multi Processor)

Il s'agit d'un système à mémoire partagée avec un nombre relativement faible de processeurs (2–16) pour l'instant

SMP

Pour éviter que le bus devienne un embouteillage, on préfère mettre un cache pour chaque processeur:



SMP

Plusieurs caches peuvent contenir une copie du même bloc

Si c'est pour lecture uniquement, il n'y a pas de problème

Pour l'écriture, il y a un problème

- Il faut éviter l'écriture simultanée par plusieurs processeurs
- Il faut informer les autres processeurs des modifications

Pour résoudre ce problème on utilise un mécanisme de furetage (snooping)

Furetage

La lecture se passe comme avant

Un processeur P qui souhaite écrire, émet un message d'invalidation sur le bus

Les autres processeurs Q1, Q2, ... vérifient s'ils ont une copie du bloc

Si oui, alors le bloc est invalidé, et le processeur Qi correspondant est forcé de le charger à nouveau s'il en a besoin

Les écritures suivantes du bloc par P n'ont pas besoin de message d'invalidation

Quand un autre processeur Qi demande le bloc, le contenu du bloc est envoyé sur le bus par P. Puis le bloc est invalidé par P

Si deux processeurs P1 et P2 souhaitent écrire simultanément, l'ordre d'exécution est déterminé arbitrairement

Synchronisation

Nécessaire avec un système multiprogrammée

Encore plus nécessaire avec un multiprocesseur

Exemple : accès au disque par un seul processus à la fois

Les processeurs modernes ont des instructions spécialisées pour la synchronisation

Une telle instruction est "swap" (échanger). Le contenu d'un registre et d'un mot en mémoire sont échangés de façon atomique

Synchronisation avec l’instruction swap

Une adresse A en mémoire sera 1 si une ressource est utilisée, 0 sinon

Pour reserver la ressource, un processeur P fait:

1. **Mettre 1 dans un registre R**
2. **SWAP A et R**
3. **Si R contient 1 alors répéter à partir de 2**
4. **Quand on arrive ici, on peut utiliser la ressource en exclusivité**

Pour libérer la ressource, il suffit de mettre 0 dans A

Synchronisation (suite)

Avec un seul processeur, chaque instruction est atomique (c’est pour ça que nous avons décidé de tenir compte des interruptions après terminaison de l’exécution de l’instruction)

Avec plusieurs processeurs, le problème est plus compliqué

Chaque processeur doit s’assurer du bus en exclusivité avant de s’en servir

C’est un problème résolu par l’arbitrage du bus (que nous n’avons pas le temps d’en parler)