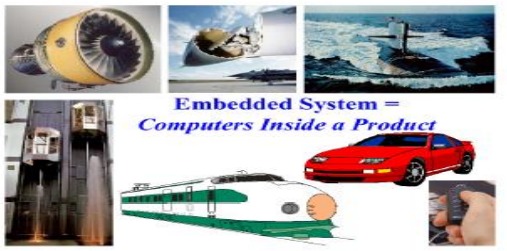
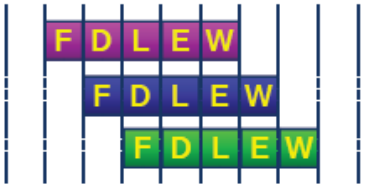
M1-SEM
Processeurs embarqués
Cours 6.1
Superscalaire-VLIW

Année 2020-2021
Pr R. BOUDOUR

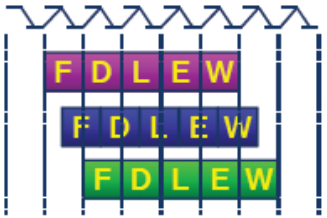


Le pipeline

Exécution séquentielle



Exécution pipelinée



Exécution superpipelinée

Principe : « Taylorisme » des instructions

Définitions :

- **Profondeur :** Nombre de subdivisions du pipeline.
- **Latence :** Temps nécessaire pour finir d'exécuter une instruction.
- **Débit :** Temps nécessaire pour exécuter deux instructions identiques sur une même unité.
- **Fréquence :** Nombre de cycle d'horloge par seconde, exprimé en hertz.
- **IPC :** Nombre d'instructions par cycle d'horloge.

Quelques problèmes & solutions

- Aléas structurels : Augmenter les ressources
- Aléas de contrôle : Prédiction de branchement
- Aléas de données : Délai + Forwarding
- Latence mémoire : Hiérarchie de caches
- Instructions multi cycles : Délai

Dans tous les cas, on peut résoudre les problèmes par des délais ➔ inefficace

Problème pipeline simple

- En pratique, les délais sont inévitables
- Un délai bloque **tout** le pipeline le temps que l'aléa soit résolu
- Un seul chemin de données dans le processeur est limitant
- **Idée :** Processeur avec plusieurs chemins de données pour pouvoir exécuter plusieurs instructions indépendantes en même temps
- **Problème :** Comment déterminer si des instructions sont indépendantes ?

Motivation

- Pour une machine pipeline, on peut espérer au mieux l'exécution d'une instruction par cycle d'horloge : il n'y a jamais plus d'une instruction lue par cycle d'horloge.
- Rappel : **temps CPU = NI * CPI * Tcycle**
- Pour augmenter la performance, on peut donc:
  - Diminuer le nombre d'instructions (NI)
  - Diminuer le nombre de cycles par instruction (CPI)
  - Diminuer la période de l'horloge (Temps de cycle)

Motivation

- Une question fondamentale se pose :
  - Est-il préférable de tenter de simplifier le circuit pour augmenter la fréquence de l'horloge ?
  - Ou bien, doit-on rendre le circuit plus complexe pour tenter de faire plus durant la même période d'horloge ?
- L'architecture RISC, avec le pipeline, procède de la première approche.
- Les résultats sont probants, mais il semble qu'on ne puisse réduire la période de l'horloge en dessous d'un certain seuil.

Il faut alors considérer la deuxième approche (ce que l'on fait avec le **VLIW et le superscalaire).**

Comment augmenter le ILP (Instruction Level Parallelism)

- En considérant plus d'une instruction par cycle d'horloge
  - L'idée de base ici est que la période de l'horloge ne peut être réduite indéfiniment, et donc que pour une certaine durée d'horloge, on doit réussir à faire plus de travail utile.
  - Comment faire ?
    - Lorsque nous arrivons à traiter tout près d'une instruction par cycle d'horloge,
    - la seule chose à faire de plus est de tenter de lire, décoder et exécuter plus d'une instruction par cycle d'horloge.

Exécution de multiples instructions par cycle d'horloge

- **Difficultés :**
  - Lorsqu'on tente d'exécuter **plus d'une instruction par cycle** d'horloge, cela implique qu'au même moment plusieurs instructions, qui ont été produites selon une certaine séquence par le compilateur et apparaissent séquentiellement en mémoire, doivent être **séparées en des chemins d'exécution parallèles**
  - **Deux façons de faire cela :**
    - Décider au moment de la compilation des instructions qui peuvent s'exécuter en parallèle (approche **VLIW**)
    - Décider au moment de l'exécution des séquences d'instructions qui peuvent se faire en parallèle (approche **Superscalaire**)

Processeurs à émissions multiples

- Les processeurs vus auparavant ne peuvent émettre au maximum qu'une instruction par cycle d'horloge : ce sont des processeurs à émission unique. Et quand on court après la performance, on en veut toujours plus : un IPC de 1, c'est pas assez ! Pour cela, les concepteurs de processeurs ont inventé des processeurs qui émettent plusieurs instructions par cycle : les processeurs à émissions multiples.
- **Ainsi :**
  - Plus de performances : CPI < 1
  - Deux solutions
    - Architecture superscalaire (SS)
    - Architecture VLIW (**V**ery **L**ong **I**nstruction **W**ord)

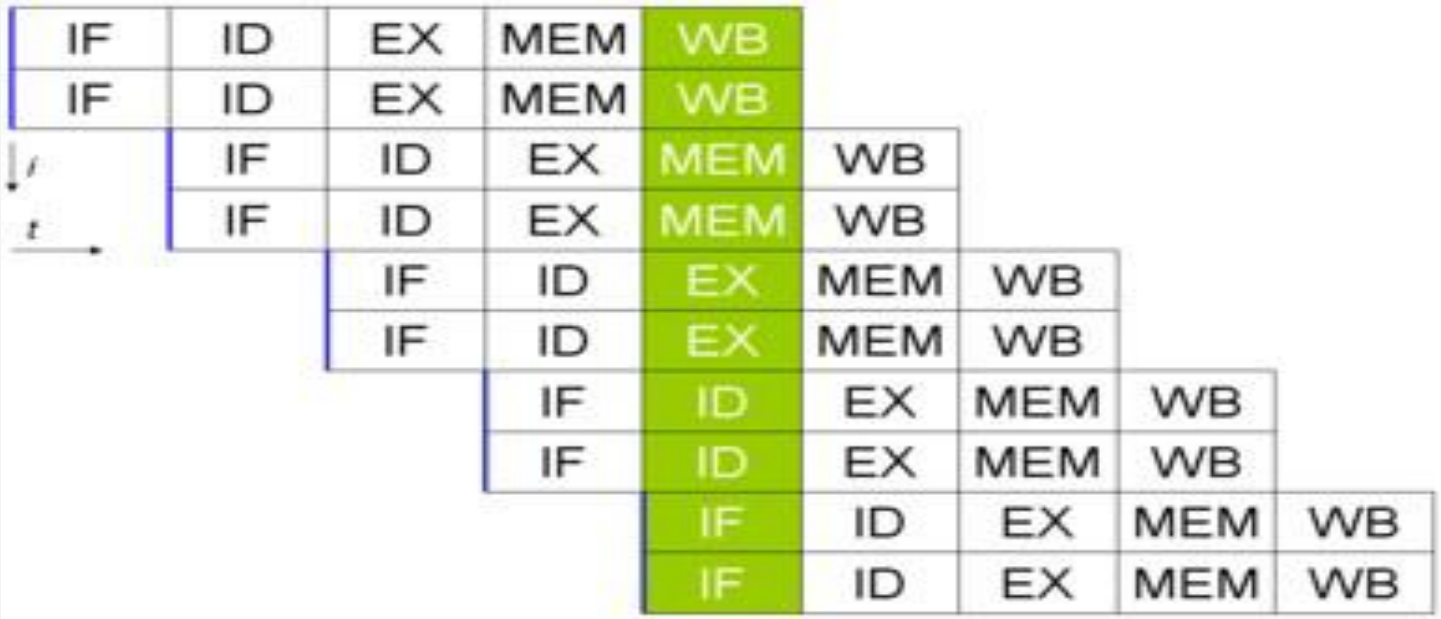
Superscalaire : qu'est-ce que c'est ?

- **Superscalaire :** technique pour exécuter plusieurs opérations en parallèle
- **Méthode :** lire en même temps des blocs d'instructions séquentielles (compilées de façon traditionnelle), et le processeur décide dynamiquement quelles instructions du bloc peuvent s'exécuter en parallèle
- **Avantage :**
  - On utilise un compilateur traditionnel pour produire le code exécutable
  - Le parallélisme est extrait de façon dynamique par le processeur, donnant une plus grande marge de manœuvre pour le scheduling des instructions

Superscalaire

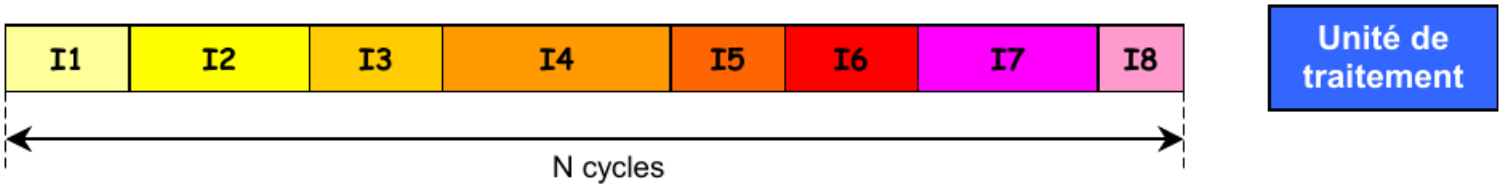
- Il faut 9 cycles pour exécuter 10 instructions.

  A t = 5, toutes les unités du processeur sont sollicitées.

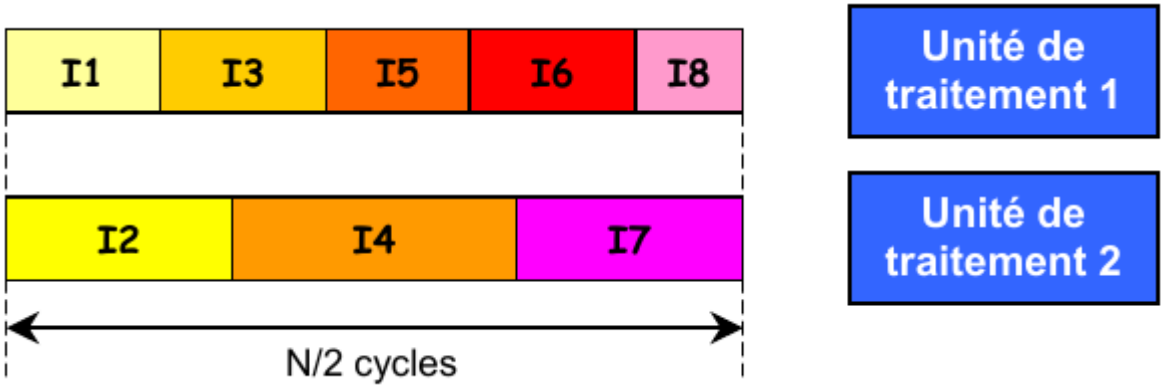


Superscalaire

❑ Architecture scalaire



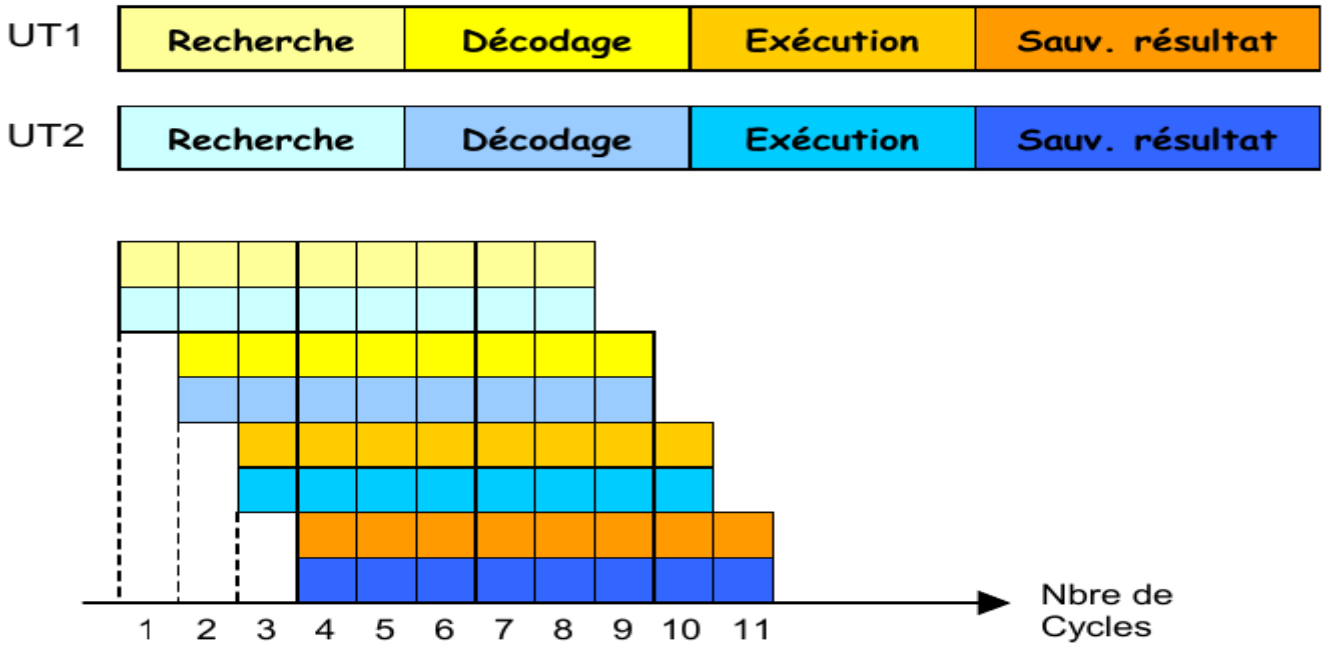
❑ Architecture superscalaire



C'est le type d'architecture mise en œuvre dans les premiers Pentium d'Intel apparus en 1993.

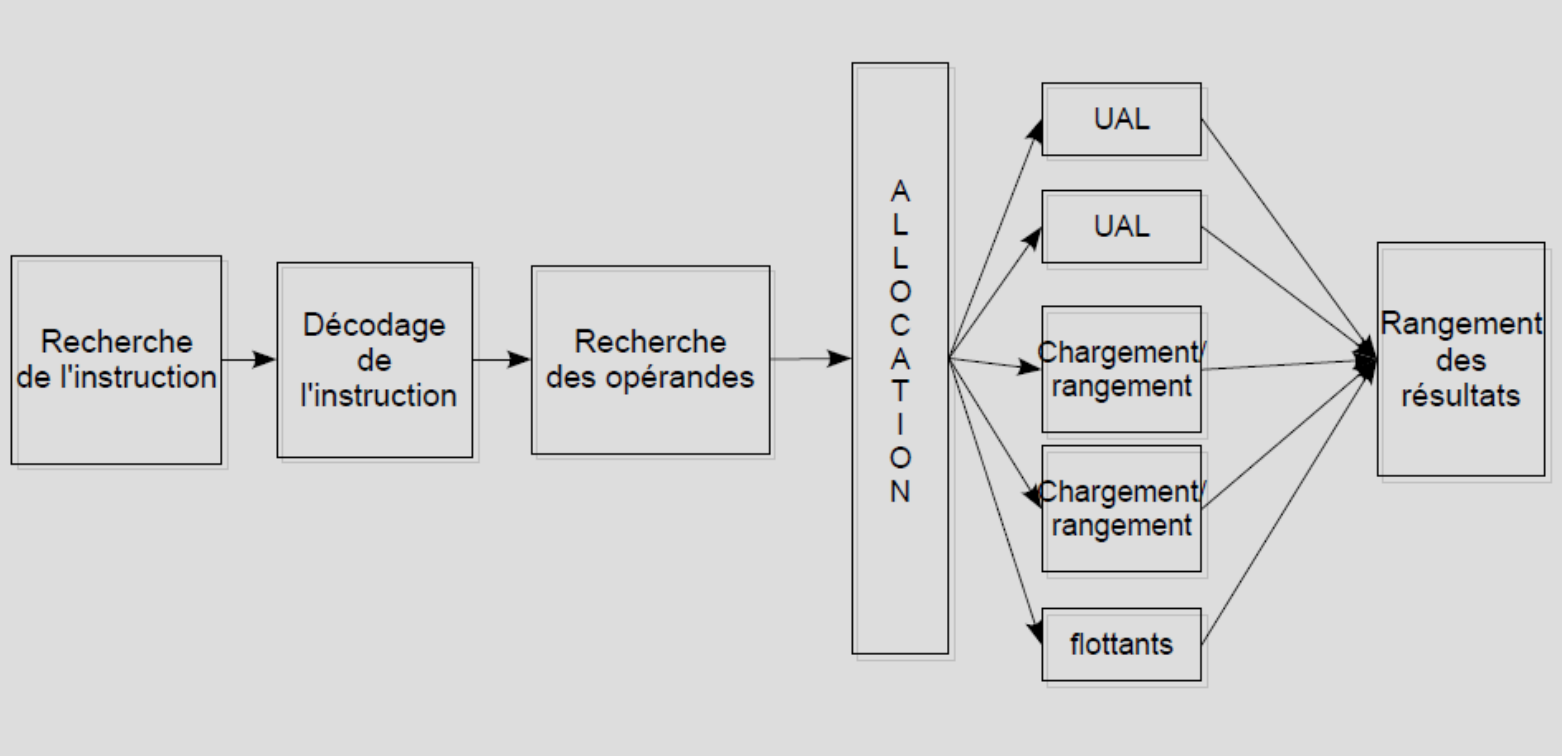
Superscalaire

- Le principe est d'exécuter les instructions de façon pipelinée dans chacune des unités de traitement travaillant en parallèle.
- **Exemple avec 4 phases et deux unités de traitement**

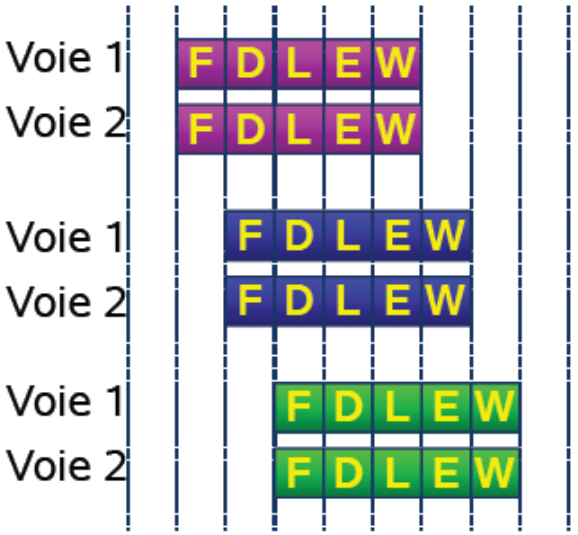


Architecture superscalaire

- **Autre exemple d'architecture SS**



Architecture superscalaire



Exécution superscalaire

- Augmente l'IPC en fonction du nombre de voies.
- Challenges
  Trouver des opérations consécutives et indépendantes
- Solutions
  - Exécution dans le désordre
  - Exécution spéculative

Problèmes liés aux superscalaires

- Nombre d'instructions lues
- Types d'instructions exécutables en parallèle
- Politique d'exécution
  - Exécution dans l'ordre
  - Exécution non ordonnée
- Dépendances de données
- Branchements
- Coût matériel accentué
  - Plusieurs accès aux bancs de registres
  - Plusieurs bus pour les opérandes et les résultats
  - Circuits d'envoi (forwarding) plus complexes
- Davantage d'unités fonctionnelles avec latences différentes
  - plusieurs fonctions (Entiers, Flottants, Contrôle)
  - plusieurs unités entières
- Plusieurs instructions exécutées simultanément
  - Débit d'instructions plus important
  - Débit d'accès aux données plus important
  - Problème des sauts et branchements accentués

Superscalaire

Processeurs superscalaires

- Dans un processeur ayant plusieurs unités d'exécution, on peut avoir plusieurs pipelines fonctionnant en parallèle. On donne le nom **superscalaire** à un processeur **capable d'émettre ainsi plus d'une instruction par cycle machine** (2 à 5)

Exemples de processeurs SS

- Le PowerPC d'IBM/Motorola/Apple,
- Le R10000 de SGI,
- L'UltraSparc de SUN,
- Le PA 8000 de HP,
- L'Alpha 21364 de DEC,
- Le RS 6000 d'IBM,
- L'Itanium d'Intel, Pentium 4, etc.

Tous sont des processeurs RISC à l'exception du Pentium (du moins vu de l'extérieur).

VLIW

- Un processeur VLIW utilise le **parallélisme d'instructions** existant dans un programme séquentiel pour démarrer à chaque cycle d'horloge l'exécution de plusieurs instructions élémentaires qui constituent le mot d'instruction très long.
- Contrairement aux processeurs superscalaires pour lesquels le matériel détermine dynamiquement les instructions exécutables simultanément, les processeurs VLIW s'appuient totalement ou principalement sur le compilateur pour constituer le mot d'instruction très long.

VLIW

- Les processeurs VLIW exécutent des **faisceaux d'instructions** (aussi appelés *bundle*), des regroupements de plusieurs instructions qui s'exécutent en parallèle sur différentes unités de calcul.
- Le faisceau est chargé en une seule fois et est encodé comme une **instruction unique**. C'est le compilateur qui regroupe plusieurs instructions en un seul faisceau.

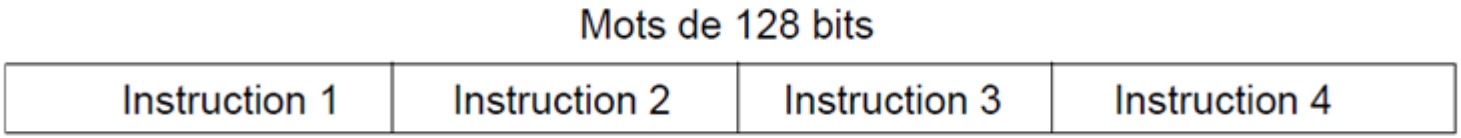
VLIW

- **VLIW :** technique pour exécuter plusieurs opérations en parallèle
- **Méthode :** on définit des **groupes de plusieurs instructions,** qui sont lues, décodées et exécutées en parallèle. Chaque groupe devient une « super-instruction » destinée à être lue d'un bloc par le processeur VLIW
- Processeur traitant des instructions de type RISC regroupées de manière à être exécutées explicitement en parallèle.
- Le VLIW est une extension relativement simple du pipeline (en théorie, on pourrait imaginer qu'on ne fait que mettre un ensemble de machines pipelines indépendantes en parallèle).
- **Avantage :**
  - le compilateur décide des instructions à grouper ensemble. Ceci élimine pour le processeur VLIW le fardeau du scheduling des instructions (détermination des instructions qui peuvent être exécutées en parallèle).
  - Le processeur est donc plus simple à concevoir, et par conséquent peut être plus rapide.

Les processeurs VLIW

- **VLIW : Very Long Instruction Word**

Le concept des VLIW :



Une instruction que l'on appelle mot est composée d'un certain nombre d'instructions élémentaires et le processeur est capable d'exécuter toutes ces instructions élémentaires en parallèle.

- Les processeurs VLIW sont utilisés dans :
  - les processeurs de **traitement du signal** (DSP),
  - ou dans les multi-coeurs
  - et «many core» pour la faible consommation.

Les processeurs VLIW

- **Compilateur détermine les opérations (VLIW)**

Operation 1	Operation 2	Operation 3	Operation 4

- **Opération dans une instruction VLIW**
  - Equivalente à une instruction superscalaire ou purement à celle d'un processeur séquentiel
  - Le nombre d'opérations dans une instruction VLIW = nombre d'unités d'exécution dans le processeur
  - Chaque opération spécifie l'instruction qui sera exécutée sur l'unité d'exécution correspondante dans un cycle indiqué de l'instruction VLIW

Les processeurs VLIW

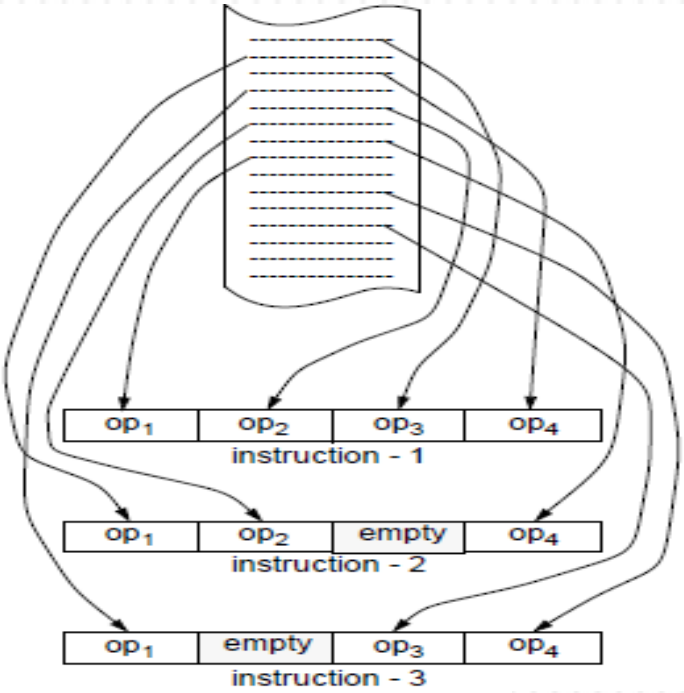
- L'approche VLIW (Very Long Instruction Word) consiste à placer dans un mot de taille assez grande (> 100 bits) plusieurs instructions prêtes à être exécutées en parallèle par des unités fonctionnelles multiples indépendantes. Un exemple de processeur de ce type est le i860 d'Intel (mots de 128 bits)
- Le parallélisme est ici géré par le compilateur et non par le processeur
- On trouvera une extrapolation de ce concept dans le Itanium d'Intel, qui utilise la technologie EPIC (Explicitly Parallel Instruction Computer). Des mots de 128 bits contiennent 3 instructions de 40 bits + des prédicats indiquant lesquelles peuvent être exécutées en parallèle.

Les processeurs VLIW

- **Cette architecture est efficace si :**
  - le compilateur extrait suffisamment de parallélisme
  - le compilateur peut mettre dans les trous du mot des instructions non operation

Processeurs VLIW

- Les architectures VLIW reposent sur détection au niveau de la compilation du parallélisme le compilateur analyse le programme et détecte des opérations pouvant être exécutées en parallèle, comme opérations pouvant être groupées dans une large instruction.
- Après que l'instruction est recherchée, toutes les opérations correspondantes sont sorties en parallèle ⇒
  - Pas de matériel requis pour la détection du parallélisme
  - Le problème de la fenêtre d'instruction est résolu : le compilateur peut analyser potentiellement le programme entier afin de détecter des opérations parallèles

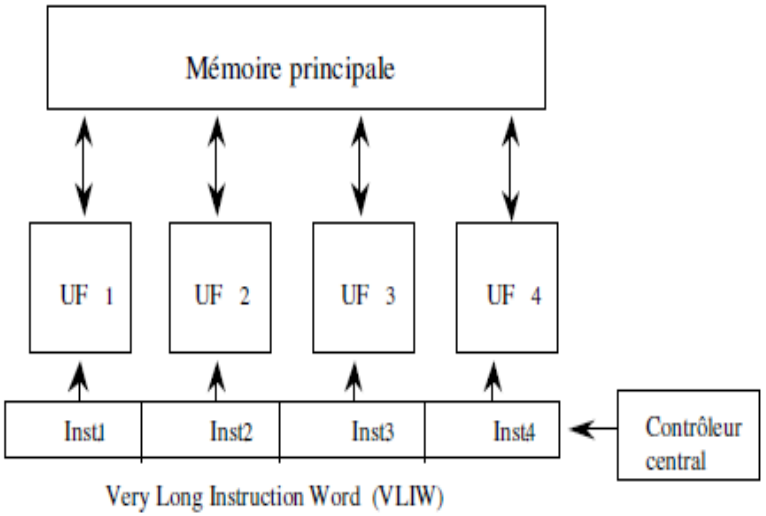


Detection of parallelism and packaging of operations into instructions is done, by the compiler, off-line.

Remarque : Très bonne performance sur des programmes écrits dans des langages séquentiels C ou Fortran quand ces programmes sont recompilés pour un processeur VLIW.

Processeurs VLIW

- PRINCIPE
  - Plusieurs unités fonctionnelles
  - ORDONNANCEMENT STATIQUE (compilateur)
    - Aléas de données
    - Aléas de contrôle
  - Exécution pipelinée des instructions
  - Exécution parallèle dans N unités fonctionnelles
- Taches du compilateur
  - Graphe de dépendances locales : Bloc de base
  - Graphe de dépendance du programme : Ordonnancement de traces
- Pour:
  - Matériel simplifié (contrôle)
- Contre:
  - La compatibilité binaire entre générations successives est difficile ou impossible



Exemple 0

- Soit un processeur VLIW avec une latence de 2 cycles pour l'instruction Load

  Soit la séquence : Add r1,r2,r3
  Ld r2, (r4)

  L'ordonnancement laisse apparaitre l'opération add dans l'instruction après Ld puisque load n'écrit dans r2 que lorsque 2 cycles sont écoulés

Le compilateur peut ordonnancer des instructions avec un aléa EAL dans le désordre : une instruction peut lire un registre avant qu'une instruction écrive dans ce registre, car l'ancienne valeur du registre n'est modifiée qu'une fois l'instruction est complètement achevée

Exemple 1 : Ordonnancement par le compilateur

- Comment ces opérations vont être ordonnancées par le compilateur ?

```
ADD r1, r2, r3
SUB r16, r14, r7
LD r2, (r4)
LD r14, (r15)
MUL r5, r1, r9
ADD r9, r10, r11
SUB r12, r2, r14
```

Hypothèses

- Nombre d'unités d'exécution = 3
- Toutes les opérations ont une latence = 2 cycles
- Toute unité d'exécution peut exécuter n'importe quelle opération

Solution

ADD *r*1, *r*2, *r*3

SUB *r*16, *r*14, *r*7

LD *r*2, (*r*4)

LD *r*14, (*r*15)

MUL *r*5, *r*1, *r*9

ADD *r*9, *r*10, *r*11

SUB *r*12, *r*2, *r*14

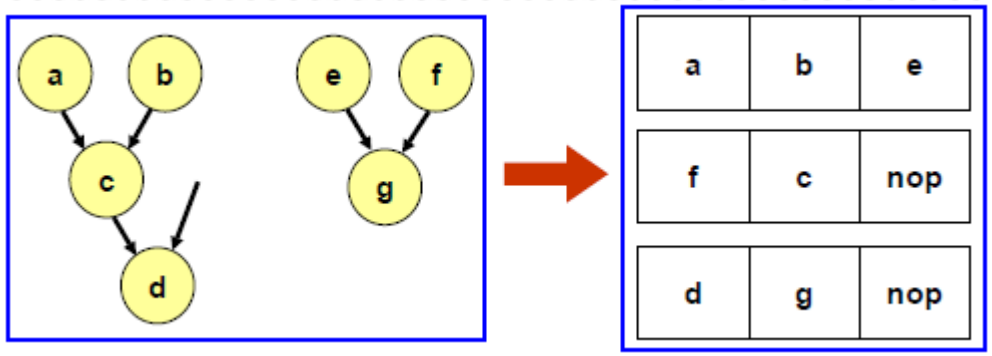
Instruction 1	ADD r_1, r_2, r_3	LD $r_2, (r_4)$	LD $r_{14}, (r_{15})$
Instruction 2	SUB r_{16}, r_{14}, r_7	ADD r_9, r_{10}, r_{11}	NOP
Instruction 3	MUL r_5, r_1, r_9	SUB r_{12}, r_2, r_{14}	NOP

Solution

- **LD r14, (r15) ordonnancée dans l'instruction avant l'opération SUB r16,r14,r7** malgré le fait que l'instruction SUB apparaît tôt dans le programme original et lit le registre de destination de LD
- Car les opérations VLIW n'écrivent pas leurs valeurs de registre qu'avant leur achèvement, la valeur ancienne de r14 reste disponible pendant deux cycles permettant à SUB de lire la valeur de r14 et de générer un résultat correct
- Ordonnancer ces opérations dans le désordre de cette façon permet au programme d'être ordonnancé avec **peu d'instructions.**

Exemple 2 :

- **Ordonnancement statique d'opérations indépendantes**



Flux de graphe de données montre les dépendances entre opérations

Sheduling and packaging des opérations dans les instructions

Exemples de processeurs VLIW

- **Les processeurs VLIW :**
  - Itanium (3 opérations)
  - Transmeta Crusoe (4 opérations)
  - Embedded processors (DSP)

Avantages et problèmes de VLIW

- **Avantages :**
  - Matériel simple
    - Le nombre d'unités foctionnelles peut augmenter sans nécessiter un matériel sophistiqué additionnel, comme SS
    - La consommation peut être réduite
  - Bons compilateurs peuvent détecter du parallélisme basé sur une analyse globale du programme (Pas de fenêtre d'instructions)
- **Problèmes :**
  - Grand nbre de registres pour maintenir les unités fonctionnelles actives
  - Besoin d'une capacité de transport de données entre unités fonctionnelles-fichier de registres et et fichier de registres-mémoire
  - Bande passante élevée entre cache instruction-unité fetch par ex : une instruction avec 7 opérations, chacune de 24 bits, 168 bits/inst
  - Code large, partiellement dû aux opérations non utilisées ⇒ perte de bits dans le mot instruction.
  - Incompatibilité du code binaire par ex: Si pour une nouvelle version de processeur, des unités fonctionnelles sont ajoutées ⇒ augmentation du nombre d'opérations possibles à exécuter en parallèle ⇒ changement du mot instruction ⇒ ancien code ne peut fonctionner sur ce processeur

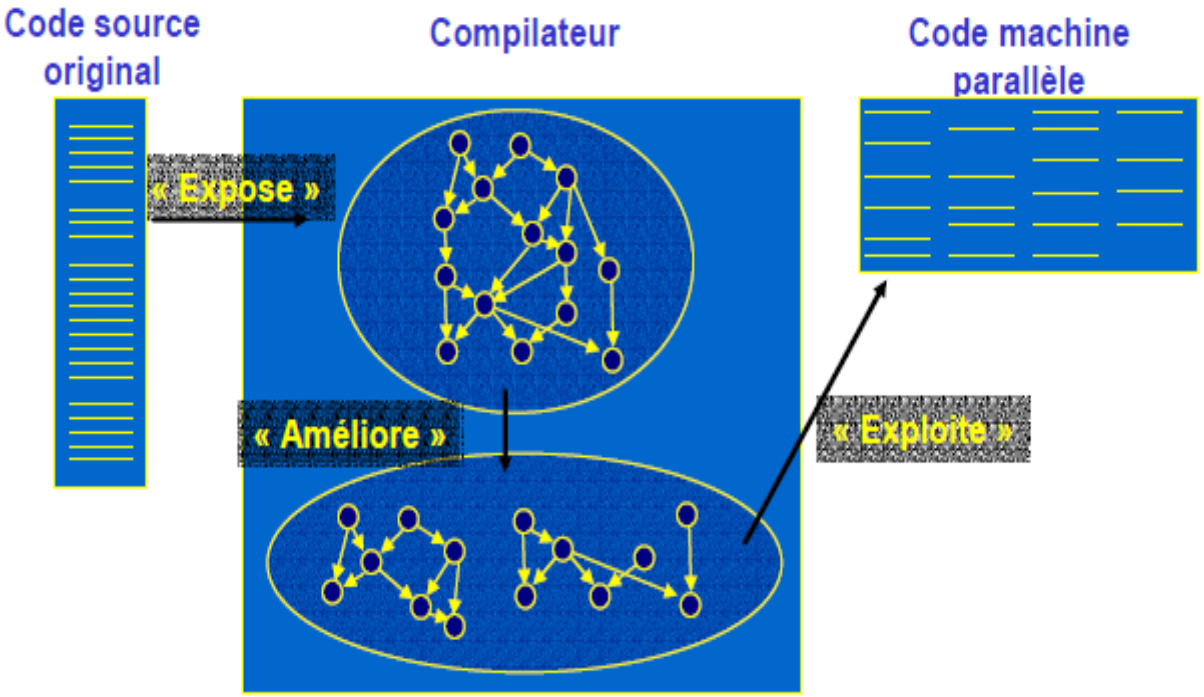
Ce qu'il faut retenir du VLIW

- VLIW processor recompile the program into the operations
- Run the operations in parallel on different execution units
- Advantage of simpler instruction issue logic
- Faster clock cycle or more execution units
- Advantage of complete control of operations and scheduling of instructions
- VLIW processor disadvantage difficulty of increasing more execution units

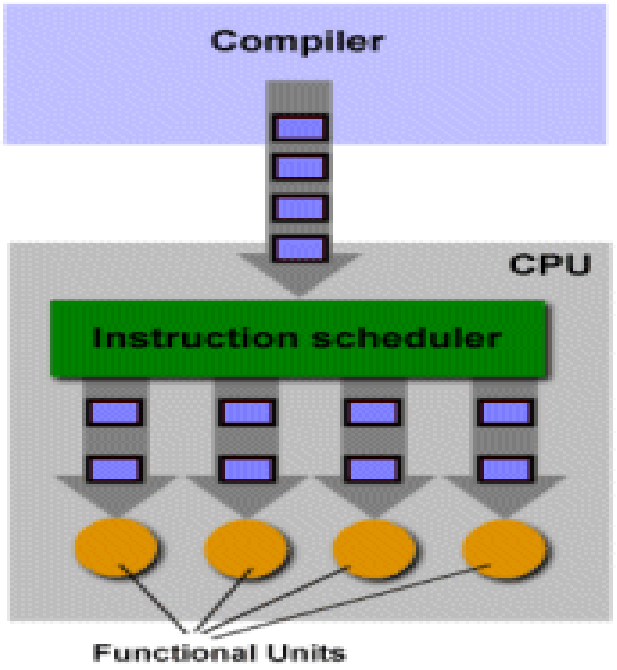
Comparaison Superscalaire-VLIW

VLIW :

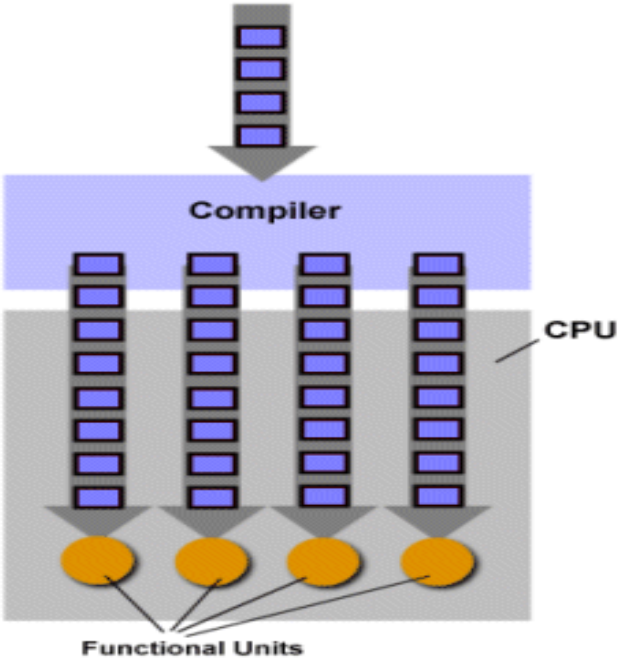
- Le compilateur expose, améliore et exploite le parallélisme du programme source et le rend explicite dans le code machine



Comparaison Superscalaire-VLIW



Dynamic Superscalar Instruction Scheduling



VLIW Instruction Scheduling

Comparaison Superscalaire-VLIW

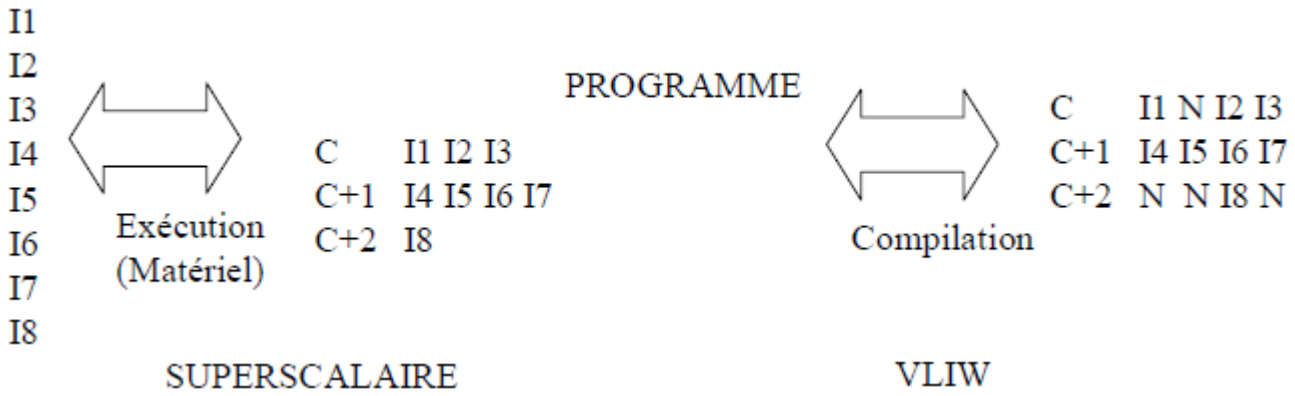
SS:

Ce qui limite la performance dans les systèmes superscalaires « standard »:

- Les branchements
  - Les mauvaises prédictions limitent la performance
  - Les petites branches (peu de code) ont peu de code à exécuter, ce qui limite le parallélisme
- L'accès à la mémoire
  - Utilise plus d'un coup d'horloge
  - Souvent, on doit faire un *load* juste après un branchement
- L'extraction du parallélisme des instructions
  - Le compilateur « sérialise » le code, dont le parallélisme intrinsèque doit être redécouvert dynamiquement par le processeur

la sélection des instructions à lancer est réalisée **dynamiquement (au cours de l'exécution)**

Comparaison Superscalaire-VLIW



- Superscalaire
  - Le **matériel** est responsable **à l'exécution** du lancement parallèle des instructions
    - Contrôle des dépendances de données et de contrôle par matériel
- VLIW
  - Le **compilateur** est responsable de présenter au matériel des instructions exécutables en parallèle
    - Contrôle des dépendances de données et de contrôle par le compilateur

VLIW vs superscalaire : les enjeux

- Le débat entre VLIW et superscalaire s'apparente au débat CISC/RISC du début des années 80 : simplicité du matériel (et performance améliorée) vs simplicité du compilateur

- Selon vous, lequel devrait gagner la bataille ?
  - VLIW ?
  - Superscalaire ?

Conclusion

- L'approche **VLIW,** basée sur un ordonnancement statique par le compilateur d'instructions dans une instruction très longue favorise l'utilisation du **pipeline logiciel,** généralement combinée avec le **déroulage de boucles** pour exécuter le maximum d'instructions en parallèle dans les différentes unités fonctionnelles. Si le contrôle des dépendances de données et de contrôle uniquement par le logiciel simplifie l'implémentation matérielle, des variantes utilisent un contrôle matériel des dépendances et des techniques comme les prédicteurs de branchement pour prendre en compte à l'exécution ce qui n'est pas connu à la compilation.
- L'approche VLIW a connu et connaît de grands succès dans les processeurs spécialisés pour le **traitement du signal** : le meilleur exemple est la série TMS320C6x de Texas Instruments. Elle est aussi beaucoup utilisée dans des coprocesseurs de systèmes sur puce (SoC) pour la partie traitement du signal. Dans ce domaine, elle combine haute performance et faible consommation.

Conclusion

- Pour les processeurs d'usage général, l'histoire du Transmeta et des processeurs Itanium a plutôt montré la supériorité des processeurs superscalaires multi-pipelines et encore plus superscalaires de type « flot de données restreint » aussi appelés à exécution non ordonnée exécutant les jeux d'instructions IA-32 et Intel64.

MERCI !

Des questions ?

Des incompréhensions ?

Questions

Q1. Qu'est-ce que le parallélisme d'instructions ? Comment les processeurs l'exploitent-ils pour améliorer les performances ?

Q2. Qu'est-ce-qu'un processeur superscalaire ?

Q3. Qu'est-ce-qu'un processeur VLIW ?

Q4. Expliquer : Réordonnancement des instructions puis exécution spéculative.

Q5. Comparer les architectures SS et VLIW ?

Réponses

- R1.
  - Le parallélisme d'instructions désigne le fait qu'un certain nombre d'instructions dans un programme séquentiel sont indépendantes, ce qui implique qu'il n'est pas nécessaire de les exécuter dans l'ordre où elles apparaissent dans le programme afin de générer le résultat correct.
  - Les processeurs exploitent cette propriété en exécutant ces instructions en parallèle plutôt qu'en séquence, ce qui réduit le temps nécessaire à l'exécution des programmes.