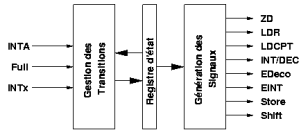
Architecture des Ordinateurs
Partie II : Microprocesseur

4. Unité de contrôle et Microprogrammation

David Simplot
simplot@fil.univ-lille1.fr

Objectifs

- Comment sont réalisées les Unités de Contrôle dans les microprocesseurs ?
  - Architecture simple (*e.g.* type RISC)
    - ⇒ implémentation câblée
    - Automate



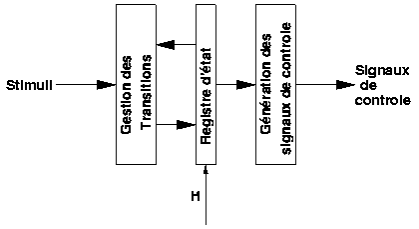
  - Architecture + complexe ???

Au sommaire...

- **Microprogrammation verticale**
- Microprogrammation horizontale
- Améliorations des performances

Microprogrammation « verticale » (1/3)

- Avec une implémentation câblée de type automate, on avait :



Microprogrammation « verticale » (2/3)

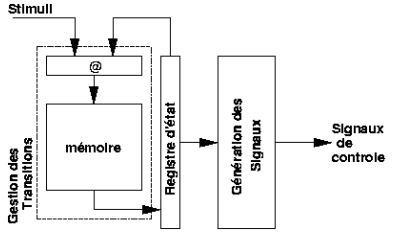
- L'idée est de faciliter l'implémentation de l'automate en stockant les transitions en mémoire…
  - On a vu dans l'exemple que
    - Nouvel état = f(état courant, stimuli)
  - État courant + stimuli forment une @ mémoire

Mémoire de type ROM interne à l'unité de contrôle !

  - A cette @ mémoire, est inscrit le nouvel état !
    - Plus rapide
    - Mais prend plus de place sur le chip

Microprogrammation « verticale » (3/3)

- Ceci donne :

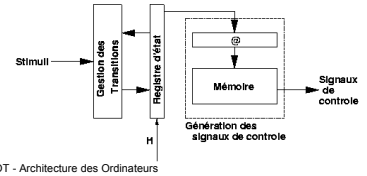


Au sommaire...

- Microprogrammation verticale
- **Microprogrammation horizontale**
- Améliorations des performances

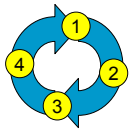
Microprogrammation « horizontale » (1/7)

- L'idée est de faire la même chose avec les signaux de contrôle générés à partir de l'état…
- On dispose d'une mémoire de micro-instructions
  - L'état de l'automate est l'adresse du microprogramme à exécuter
  - On parle de pointeur de microprogramme et non plus d'état de l'automate



Microprogrammation « horizontale » (2/7)

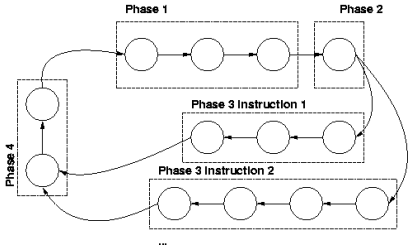
- Dans un microprocesseur, la plupart des chemins des chemins dans l'automate sont des séquences
- Le seul choix est fait lors du décodage de l'instruction (phase 2)



1. Lire
2. Décoder
3. Exécuter
4. Préparer

Microprogrammation « horizontale » (3/7)

- Automate de l'unité de contrôle d'un microprocesseur :



Microprogrammation « horizontale » (4/7)

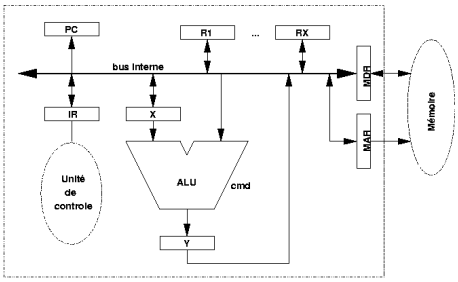
Chaque ligne de la mémoire de microprogramme est une micro-instruction

- Directement reliée aux transistors générant les signaux de contrôle
- Implicitement, la micro-instruction suivante est à l'adresse suivante
- On a une colonne particulière pour dire que c'est au décodeur d'instruction de générer l'adresse de la micro-instruction suivante

Microprogrammation « horizontale » (5/7)

- Les séquences de micro-instructions sont celles que l'on a vu dans le chapitre 2 :
  - Phase 1 :
    - PCout – LDMAR – LDX
    - Read – INCX – LDY
    - Yout – LDPC – WaitMemory
    - MDRout – LDIR
  - Phase 2 :
    - Mettre dans le pointeur de micro-programme l'@ de la séquence correspondant à l'instruction

Microprogrammation « horizontale » (6/7)



Microprogrammation « horizontale » (7/7)

- Phase 3 : (exemple pour MOV R1, AA)
  - 3.1 lecture argument + incrémentation PC
    - PCout – LDMAR – LDX
    - Read – INCX – LDY
    - Yout – LDPC - WaitMemory
  - 3.2 exécuter l'instruction
    - R1out – LDX
    - MDRout – ADD – LDY
    - Yout – LDR1

Au sommaire...

- Microprogrammation verticale
- Microprogrammation horizontale
- **Améliorations des performances**

Améliorations des performances (1/15)
Mesure des performances

- Performance = vitesse de traitement

Temps/tâche
=
instructions/tâche
x
cycles/instruction
x
temps/cycle

Améliorations des performances (2/15)
Mesure des performances (suite)

Temps/tâche=instructions/tâchexcycles/instructionxtemps/cycle

- Instructions/tâche
  - Dépend :
    - Du jeu d'instructions (RISC/CISC)
    - Algorithme pour réaliser la tâche
    - Niveau d'optimisation du compilateur
- Cycles/instruction
  - Dépend :
    - De la complexité des instructions utilisées (RISC/CISC)
    - Optimisation du compilateur (choix des instructions)

Améliorations des performances (3/15)
Mesure des performances (suite)

Temps/tâche=instructions/tâchexcycles/instructionxtemps/cycle

- Temps/cycle
  - Directement dérivé de la fréquence de l'horloge
  - Dépend :
    - Complexité de l'architecture
    - Technologie

Améliorations des performances (4/15)
Philosophies CISC/RISC

- CISC
  - Jeu d'instructions avec un grand nombre d'instructions
    - *E.g.* instructions MMX
  - de nombreux modes d'adressage
    - *E.g.* adressages indexés… la plupart des instructions peuvent adresser la mémoire
  - Soucis de compatibilité avec les générations précédentes
    - « compatibilité ascendante »
- RISC (début des années 80 CRAY/IBM)
  - Jeu d'instructions limité dans le but de minimiser le temps d'exécution
    - *E.g.* seuls les instructions LOAD et STORE adressent la mém.

Améliorations des performances (5/15)
Philosophies CISC/RISC (suite)

Caractéristique	RISC	CISC
Nbre d'instructions	<100	>200
Nbre de modes d'adressage	1 à 2	5 à 20
Nbre de format d'instructions	1 à 2	3+
Nbre cycles/instructions	~1	3 à 10
Accès à la mémoire	load/store	~ toutes
Nbre registres	32+	2 à 16
Réalisation µP	câblé	µ-pgm
Logique pour décodage	10%	50%

 Courtesy René Chevance/CNAM

Améliorations des performances (6/15)
Convergence RISC/CISC

- Convergence des performances CISC/RISC
  - La plupart des technologies RISC sont reprises dans les architectures CISC.
- Certains microprocesseurs CISC (*e.g.* Pentium II+) traduisent les instructions en suite de micro-instructions et fonctionnent comme un processeur RISC sur ces micro-instructions
  - Implémentation câblée

Améliorations des performances (7/15)
Parallélisation des micro-instructions

- Dans l'écriture des séquences de micro-instructions correspondant à une instructions
  - On a mis en // différentes micro-instructions.
- En prenant plusieurs instructions d'un coup (typiquement 3 ou 4), on peut paralléliser les différentes micro-instructions
  - Pb. Des branchements peuvent intervenir et rendre faux les opérations déjà exécuter
  - ⇒ anticipation de branchement (branchement prédictifs)
  - ⇒ il faut pouvoir annuler une opération
  - « Out of Order »

Améliorations des performances (8/15)
Parallélisation des micro-instructions (suite)

- Out of Order
  - Vient des CPU « super-scalaires »

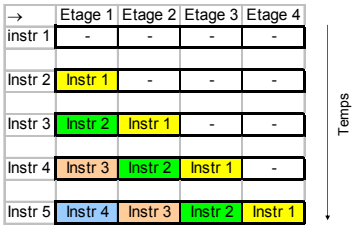
```
1: Add r1, r2 -> r8
2: Sub r8, r3 -> r3
3: Add r4, r5 -> r8
4: Sub r8, r6 -> r6
```

  - Les instructions 1 et 3 peuvent être exécutées en parallèle si r8 est renommé

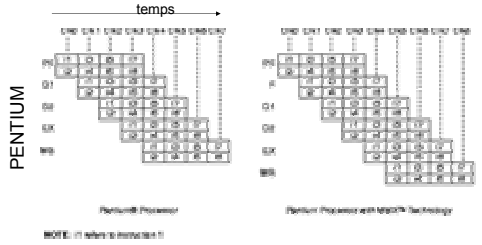
```
Add r1, r2 -> r8   |   Add r4, r5 -> r9
Sub r8, r3 -> r3   |   sub r9, r6 -> r6
```


Améliorations des performances (9/15)
Parallélisation des micro-instructions (suite)

- Technique RISC : « pipe-line »
  - Microprocesseur en étage (travail à la chaîne)

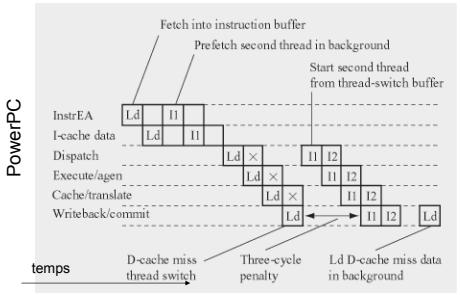


Améliorations des performances (10/15)
Parallélisation des micro-instructions (suite)



- PF=Prefetch, F=Fetch, D1=Instruction decode, D2=Address Generate, EX=Execute, WB=Writeback

Améliorations des performances (11/15)
Parallélisation des micro-instructions (suite)



Améliorations des performances (12/15)
Architecture IA-64

- Accord entre Intel et HP pour mettre au point une architecture 64 bits haut de gamme
- IA-32 + HP PA ⇒ IA-64 (Itanium)
- Architecture « CISC » ?
  - Multi-étages (une dizaine)
  - Compatible IA-32 avec génération de micro-code parallélisé à la volée (RISC)
    - Défauts de l'IA-32 : nbre de registres limité + calcul flottant lents + capacité mémoire limitée à 4 Go ☺
  - Mode IA-64
    - EPIC : Explicitly Parallel Instruction Computing

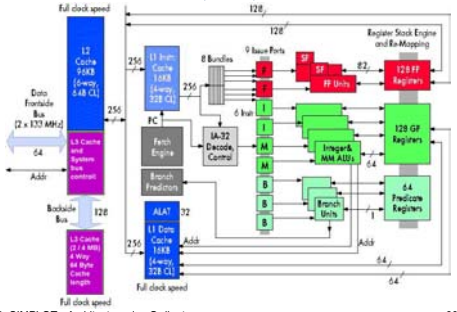
Améliorations des performances (13/15)
Architecture IA-64 (suite)

- Techniques utilisées :
  - Parallélisation des instructions
  - Anticipation de branchements
    - Ne pas briser le pipe-line et les instructions en cours
  - Prédiction des chargements
    - Les chargements en mémoire sont très pénalisants et le sont de plus en plus avec l'accélération des µP
- Nombreux registres 64 bits (128 !)
- Les instructions sont regroupées en paquets (*bundle*) de 3 instructions (certains disent de 1 à 9+ ?)
  - Descripteur permettant d'anticiper et vérifier que les chargements ont été fait (prédiction des chargements) ainsi que tester s'il faut exécuter le bloc (anticipation des branchement)

Améliorations des performances (14/15)
Architecture IA-64 (suite)

- 10 étages de l'itanium :

Améliorations des performances (15/15)
Architecture IA-64 (suite)



Conclusion

- On a vu :
  - Architecture interne d'un microprocesseur
  - Les chemins de données
  - La mémoire
  - La gestion des périphériques avec les interruptions et les DMA
  - Les optimisations possibles
- Reste à voir :
  - Liens entre matériel et logiciel (Partie III)
    - Système d'exploitation, génération de code,…
  - Gestion de la mémoire, des entrées/sorties (Partie IV)