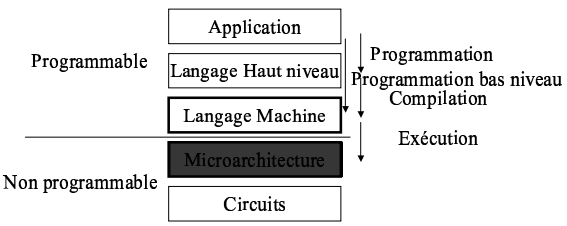
Architecture des ordinateurs
Partie 2

IFIPS Cycle Apprentissage
2005-2006

Cécile Germain-Renaud
cecile.germain@lri.fr

3. Micro-architecture

Les niveaux d'un système informatique



Objectif

- Comment réaliser un micro-processeur très simple à l'aide de circuits élémentaires

Plan

- Algèbre de Boole
- Circuits combinatoires
- Circuits séquentiels
- Partie opérative
- Partie contrôle

Introduction

- Information Etat Haut - Etat Bas
- Abstraction :
  - Etat Haut -> 1
  - Etat Bas -> 0

Propriétés formelles de la représentation abstraite en vue d'une réalisation matérielle

Algèbre de Boole

Soit *E* un ensemble non vide, muni de deux opérations binaires + et ., d'une opération unaire notée $^-$, et deux éléments particuliers de E notés 0 et 1. (*E*, 0, 1, +, ., $^-$) est une algèbre de Boole si

(i) les opérations + et . sont commutatives et associatives

(ii) 0 est neutre pour + et 1 pour .

(iii) + et . sont distributives l'une sur l'autre.

(iv)

$$\forall a \in E, a.\bar{a} = 0 \text{ et } a + \bar{a} = 1$$

L'algèbre de Boole des circuits logiques

• E = {0, 1} et +, . , $^-$ sont définies formellement par :

a	b	a + b	a.b
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

a	$\bar{a}$
0	1
1	0

Interprétation logique, au sens philosophique,

. = ET, + = OU, $^-$ = NEGATION (NOT)

Propriétés

- Notation : priorité de . sur +

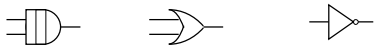
- $\bar{\bar{a}}$ = a
- a + 1 = 1 a.0 = 0
- a + 0 = a a.1 = a
- a + a = a a.a = a

- Règles de De Morgan

$$\forall a,b \in E, \overline{a+b} = \bar{a}.\bar{b}$$
$$\forall a,b \in E, \overline{a.b} = \bar{a} + \bar{b}$$

Portes logiques

Porte logique : circuit qui réalise une fonction élémentaire



ET OU Inverseur

Portes NAND et NOR

- Les portes ET et OU ne sont pas les plus simples à réaliser en technologie CMOS
- Les "vraies" portes élémentaires sont les portes NAND, NOR et NOT

$$NAND(a,b) = \overline{a.b}$$

$$NOR(a,b) = \overline{a + b}$$

$NAND_k$ et NOR_k

$$NAND_k(x_1, \ldots, x_k) = \overline{x_1.x_2. \ldots .x_k}$$

$$NOR_k(x_1, \ldots, x_k) = \overline{x_1 + x_2 + \ldots + x_k}$$

NOR4 NAND4

Mais les contraintes technologiques limitent à un petit nombre d'entrées.

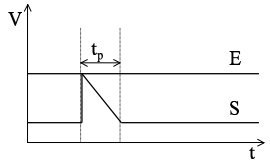
Le OU exclusif

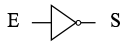
a	b	a ⊕ b
0	0	0
0	1	1
1	0	1
1	1	0

$$a \oplus b = \bar{a}.b + a.\bar{b}$$

Portes logiques et algèbre de Boole

- Les portes logiques ne réalisent les fonctions logiques qu'à certains points dans le temps
- Le *temps de propagation* d'un circuit est le temps qui sépare le positionnement des entrées de celui de la sortie





Fonctions booléennes

Fonctions de B^n vers B

Représentations :

En extension : tables de vérité

Algébrique : f(a,b,c) = b + a.c

La représentation algébrique n'est pas unique :

a + a = a

a	b	c	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Représentation des fonctions booléennes

- Deux représentations canoniques
  - Forme conjonctive normale et forme disjonctive normale
  - Une fonction booléenne a une et une seule forme conjonctive normale, et une et une seule forme disjonctive normale
  - Technologiquement utilisables
- Formes minimisées suivant des critères technologiques

Forme disjonctive normale (FDN)

But : exprimer la fonction comme un polynôme aux x_i et $\overline{x}_i$

Théorème de Shannon :

$f(x_1, \ldots x_n) = x_1 . \overline{f(0}, x_2, \ldots x_n) + x_1 . f(1, x_2, \ldots x_n)$

Les deux membres sont identiques pour $x_1 = 0$ et $x_1 = 1$

Forme disjonctive normale

Un terme produit (minterm) est un produit de toutes les variables d'entrées, complémentées ou non.

Pour n variable d'entrée, il y a 2^n minterms

Chaque minterm correspond à une ligne de la table de vérité. Chaque variable est complémentée si dans cette ligne sa valeur est 0, non complémentée sinon

$m_0 = \overline{x}_1 . \overline{x}_0$
$m_1 = \overline{x}_1 . x_0$
$m_2 = x_1 . \overline{x}_0$
$m_3 = x_1 . x_0$

x1	x0	m0	m1	m2	m3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Forme disjonctive normale

La forme disjonctive normale d'une fonction est le OU des minterms pour lesquels la fonction vaut 1

a	b	c	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

$$f = \bar{a}.\bar{b}.\bar{c} + \bar{a}.\bar{b}.c + a.\bar{b}.\bar{c} + a.\bar{b}.c + a.b.c$$

FDN et portes logiques

La forme disjonctive normale peut aussi s'exprimer par un NAND de NAND, en utilisant éventuellement des portes $NAND_k$

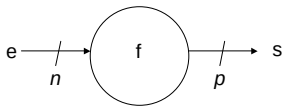
$$\begin{aligned} f &= \bar{a}.\bar{b}.\bar{c} + \bar{a}.\bar{b}.c + a.\bar{b}.\bar{c} + a.\bar{b}.c + a.b.c \\ &= \overline{\bar{a}.\bar{b}.\bar{c} + \bar{a}.\bar{b}.c + a.\bar{b}.\bar{c} + a.\bar{b}.c + a.b.c} \\ &= \mathrm{NAND}_4(\mathrm{NAND}_3(\bar{a}.\bar{b}.\bar{c}), \mathrm{NAND}_3(\bar{a}.\bar{b}.c), \\ &\qquad \mathrm{NAND}_3(a.\bar{b}.\bar{c}), \mathrm{NAND}_3(a.\bar{b}.c), \mathrm{NAND}_3(a.b.c)) \end{aligned}$$

Plan

- Algèbre de Boole
- Circuits combinatoires
- Circuits séquentiels
- Partie opérative
- Partie contrôle

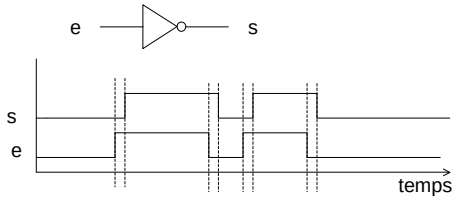
Définition

- Circuit combinatoire : dispositif matériel qui calcule = réalise une ou des fonctions booléennes
  - $S = (f_0(e_0, e_1, \ldots e_{n-1}), \ldots, f_{p-1}(e_0, e_1, \ldots e_{n-1}))$
- Synthèse logique : méthodes de conception des circuits combinatoires
- Compromis temps de propagation / encombrement / facilité de conception



Comportement temporel

- Les entrées suivent les sorties - avec un temps de retard



Typologie

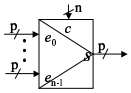
- Les fonctions élémentaires
- Les fonctions quelconques
- L'Unité Arithmétique et Logique

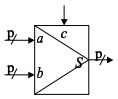
Les fonctions élémentaires

- Les opérateurs logiques : par les portes ET, OU, NAND, NOR, XOR…
- Portes complexes, dépendant de la technologie
- La sélection : par le multiplexeur ou le décodeur

Les Multiplexeurs

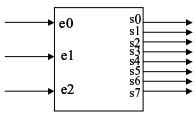
- 2^nxp entrées de données, n entrées de contrôle, 1xp sortie
  $S = e_i$ si $c_{n-1}…c_0 = i$
- Fonction : sélection
  le + simple : 1 parmi 2
  si c = 0 alors a sinon b
  Si p = 1, 1 porte XOR





Les décodeurs

- n entrées de données, 2^n sorties = les minterms. Donc, 1 seule sortie active un instant donné
  $S_i = 1$ si $e_{n-1}…e_0 = i$

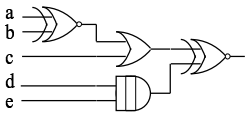


$$S_1 = \overline{e_2}.\overline{e_1}.e_0$$

Fonctions booléennes quelconques

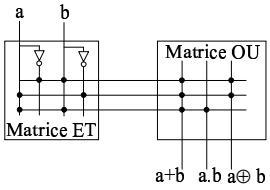
- Une fonction de quelques variables, pas de régularité : logique "anarchique"

$$(a \oplus b + c) \oplus (d.e)$$



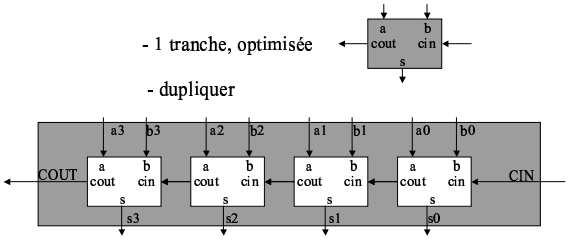
Fonctions booléennes quelconques

- Plusieurs fonctions de plusieurs variables, pas de régularité : Programmable Logic Array (PLA)
- Optimise la surface et la faisabilité



Fonctions booléennes quelconques

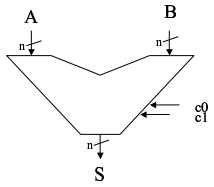
- *n*x1 fonction de plusieurs variables : logique en tranches



L'UAL

- Addition, soustraction, opérations logiques bit à bit
- Implique entrée de sélection : commande

c1	c0	S
0	0	A AND B
0	1	A OR B
1	0	A + B
1	1	A - B

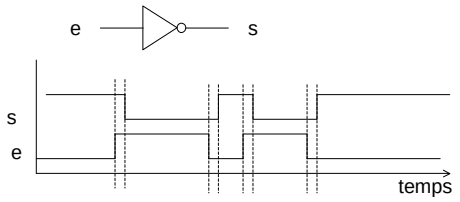


Plan

- Algèbre de Boole
- Circuits combinatoires
- Circuits séquentiels et automates
- Partie opérative
- Partie contrôle

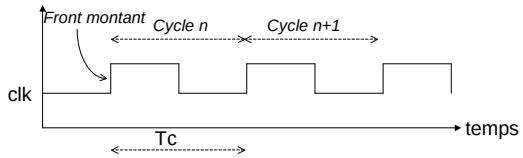
Comportement temporel

- Les entrées suivent les sorties - avec un temps de retard



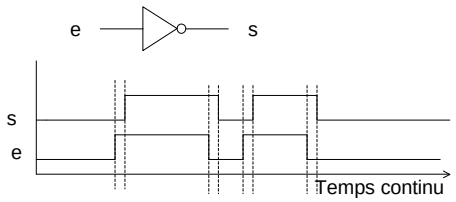
Horloge

- Signal périodique. Période = Temps de cycle T_c
  Fréquence $F = 1/T_c$
- Fournit une référence de temps discret commune à un ensemble de circuits



Objectif

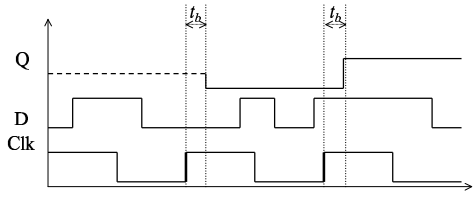
Définir les valeurs observables dans les circuits



La bascule D

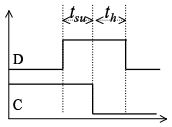
Bascule sur front

- Enregistre l'état de D sur la transition montante de l'horloge
- L'état est recopié sur Q après t_b.



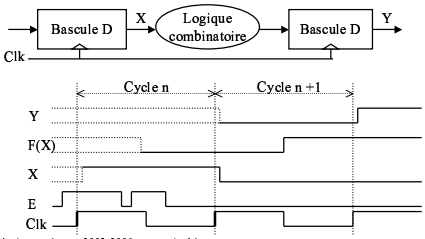
La bascule D

- Contraintes : temps d'établissement et de maintien



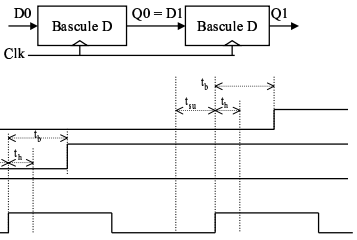
Opérateurs synchrones

Y(cycle n+1) = X (cycle n)



Exemple : Décaleur

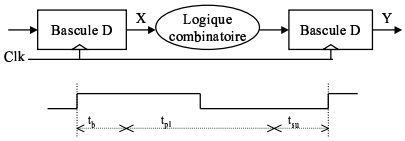
Q1(cycle n+1) = Q0 (cycle n)



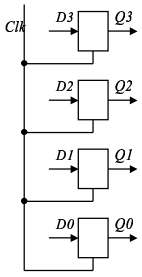
Fréquence de fonctionnement

$$T_c \geq t_b + t_{pl} + t_{su} = T_{c\min}$$

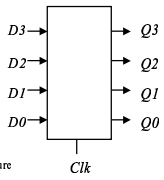
$$F_{\max} = \frac{1}{T_{c\min}}$$



Registre



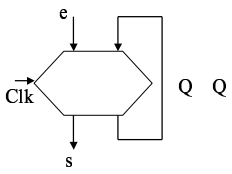
Un ensemble de bascules D commandées par le même signal : mémorisent simultanément *n* bits



Automates synchrones

$(Q_{n+1}, s) = f(Q_n, e)$

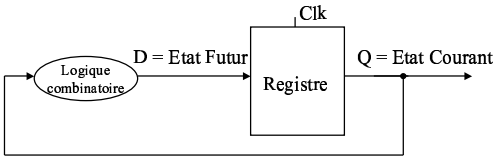
- Q : état de l'automate
- e : entrées
- s : sorties - commandes



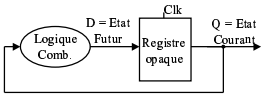
Compteurs synchrones

$X_{i+1} = 1+X_i \mod N$

Un compteur est un automate : e absent, s = Q



Compteur par 4



En code entier naturel

Etat Courant			Etat Futur		
Val	Q1	Q0	D1	D0	Val
0	0	0	0	1	1
1	0	1	1	0	2
2	1	0	1	1	3
3	1	1	0	0	0

$D0 = \overline{Q0}$

$D1 = Q1.\overline{Q0} + \overline{Q1}.Q0 = Q1 \oplus Q0$

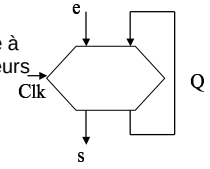
Automates synchrones

$(Q_{n+1}, s) = f(Q_n, e)$

Applications :

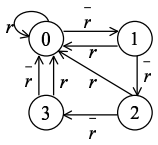
Circuits : de l'automate de machine à laver au contrôle de microprocesseurs

Théorie et pratique des langages



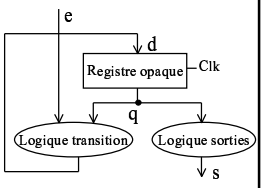
Exemple minimal

- Compteur par 4 avec RAZ



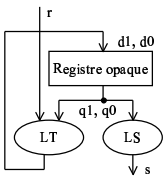
Réalisation d'un automate

- Etat : registre
- Transitions et sortie : logique combinatoire
- Réaliser un automate :
  - Taille du registre d'états
  - Synthèse des fonctions booléennes transition et sortie : PLA



Compteur avec RAZ

Etat courant				Etat futur		
N	q1	q0	r	d1	d0	N
0	0	0	0	0	1	1
0	0	0	1	0	0	0
1	0	1	0	1	0	2
1	0	1	1	0	0	0
2	1	0	0	1	1	3
2	1	0	1	0	0	0
3	1	1	0	0	0	0
3	1	1	1	0	0	0



$$d_0 = \overline{q_0}.\overline{r}$$

$$d_1 = \overline{q_1}.q_0.\overline{r} + q_1.\overline{q_0}.\overline{r}$$

$$s_1 = q_1, \quad s_0 = q_0$$

Plan

- Algèbre de Boole
- Circuits combinatoires
- Circuits séquentiels et automates
- Partie opérative (ou chemin de données)
- Partie contrôle

Organisation générale

RI <- Mem[PC] et PC <- PC + 4

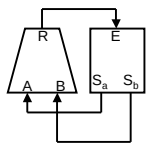
Exécution de l'instruction

- Partie opérative : transferts registre-registre, registre-mémoire, calculs.
- Partie contrôle : séquencement des actions élémentaires

Inst. arithmétiques et logiques – F1

Spécification : Rd <- Ra OP Rb

Réalisation
Rd <- Ra OP Rb



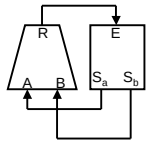
$T_c >= T_{reg} + T_{UAL}$

Inst. arithmétiques et logiques – F1

Rd <- Ra OP Rb

Rd <- Ra OP Rb

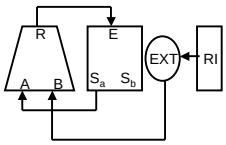
- Sélection de OP : voir partie contrôle
- Sélection de Ra, Rb, Rd : voir partie contrôle



Inst. arithmétiques et logiques – F2

Spécification : Rd <- Ra OP ES(Imm_{16})
Ou
Rd <- Ra OP (0x0000||Imm_{16})

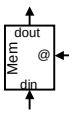
Réalisation
Rd <- Ra OP EXT(Rb)



$T_c >= T_{reg} + T_{UAL}$

La mémoire

- Lecture et Ecriture sont mutullement exclusives
- $T_{mem} = T_{UAL}$

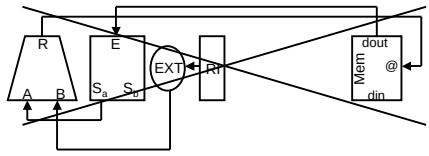


Instruction de chargement - LOAD

Spécification : Rd <- Mem [Ra +ES(Imm_{16})]

Réalisation 1 : Rd <- Mem [Ra +EXT(RI)]

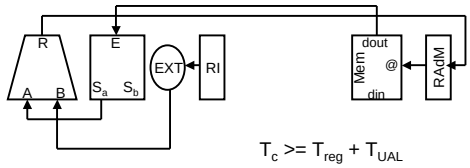
$T_c >= T_{reg} + T_{UAL} + T_{Mem}$



Instruction de chargement - LOAD

Spécification : Rd <- Mem [Ra +ES(Imm_{16})]

Réalisation 2 :
Cycle 1 : RAdM <- Ra +EXT(RI)
Cycle 2 : Rd <- Mem[RAdM]

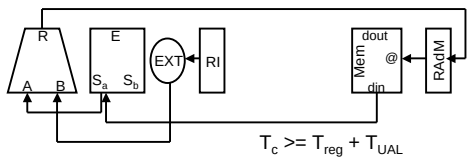


$T_c >= T_{reg} + T_{UAL}$

Instruction de rangement - STORE

Spécification : Mem [Ra +ES(Imm_{16})] <- Rd

Réalisation 2 :
Cycle 1 : RAdM <- Ra +EXT(RI)
Cycle 2 : Mem[RAdM] <- Rd

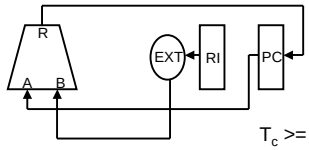


$T_c >= T_{reg} + T_{UAL}$

Instructions de branchement - Bcond

Spécification : PC <- PC + ($dep_{27}||00_b$) si cond
Autres que BA : voir partie contrôle

Réalisation
Rd <- Ra OP EXT(Rb)



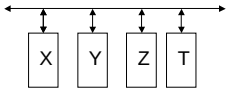
$T_c >= T_{reg} + T_{UAL}$

Les bus (1)

- Bus = ensemble de 32 « fils » interne au micro-processeur
- Pas de protocole : très différent d'un bus de carte (SCSI, USB,…)
- Pas de mémorisation
- **Contrainte : UNE SEULE information à la fois sur un bus**

Les bus (2)

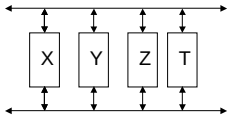
- **Contrainte : UNE SEULE information à la fois sur un bus**



- Copier X dans Y et Z : 1 cycle
  Z <- X; Z <- Y
- Copier X dans Y et Z dans T : 2 cycles
  Cycle 1 : Y <- X
  Cycle 2 : T <- Z

Les bus (3)

- **Contrainte : UNE SEULE information à la fois sur un bus**



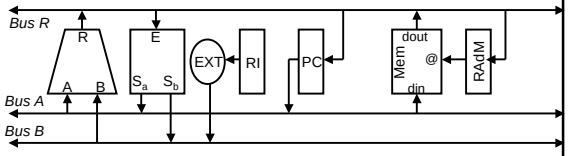
• Copier X dans Y et Z dans T : 1 cycle
Y <- X ; T <- Z

Langage transfert

- Description de la partie opérative
- **Contraintes :**
  - **UNE SEULE information à la fois sur un bus**
  - **Par des connexions existantes**
  - **Entre éléments de mémorisation : Registre-registre ou registre mémoire**
- Syntaxe : une ligne par cycle – transferts simultanés séparés par « ; »

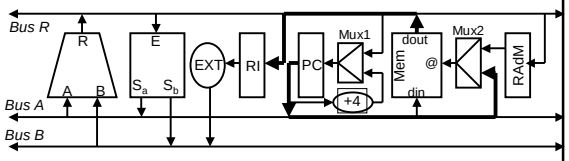
Architecture 3 bus

- A vérifier en langage transfert



Lecture de l'instruction

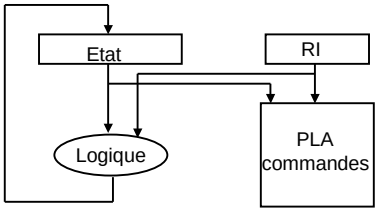
- RI <- Mem[PC] ; PC <- PC +4



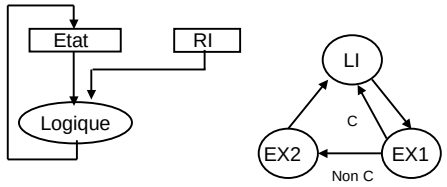
Plan

- Algèbre de Boole
- Circuits combinatoires
- Circuits séquentiels et automates
- Partie opérative (ou chemin de données)
- Partie contrôle

Organisation



Automate d'états



C = Instruction arithmétique et logique
ou branchement

4. Hiérarchie Mémoire

Plan

- Motivations
- La Mémoire Centrale
- La Mémoire Virtuelle

Plan

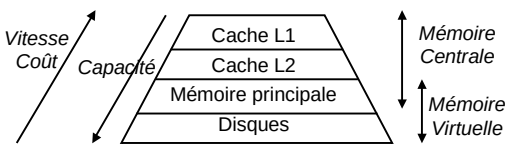
- Motivations
- La Mémoire Centrale
- La Mémoire Virtuelle

Plan

- Motivations
- Cache
- Mémoire virtuelle

Hiérarchie mémoire

- Machine de Von-Neumann : mémoire RAM, tableau d'octets
- Réalisation matérielle



Faits technologiques

- Mémoire statique - SRAM
  - technologie semiconducteurs logique
  - 6 T par point mémoire
  - Evoluent en fréquence comme les processeurs
- Mémoire dynamique – DRAM
  - Technologie semiconducteurs analogique
  - 1,5 T par point mémoire
- Disques
  - Technologie magnétique

Faits technologiques

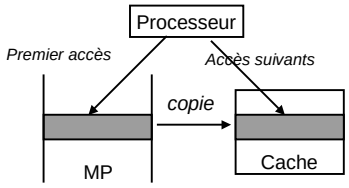
- Mémoires SRAM : temps d'accès
- Mémoire DRAM : temps d'accès et temps de cycle
- Les meilleures SRAM
  temps d'accès = Tc processeur
- 1 ordre de grandeur SRAM -> DRAM
  S'aggrave avec le temps
- 3 ordre de grandeur DRAM -> disques

Hiérarchie mémoire

- A un instant donné, chaque niveau de la hiérarchie contient un extrait du niveau inférieur
- L'accès étant plus rapide, la copie offre l'illusion d'une mémoire + rapide à faible coût
- La gestion des copies et problèmes reliés doit être transparente au niveau de l'exécution des instructions
  - Cache : entièrement en matériel
  - Mémoire virtuelle : matériel et logiciel système

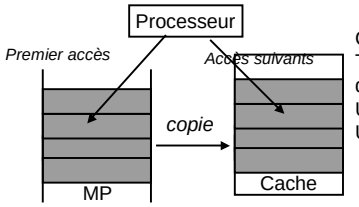
Le principe de localité

- Localité temporelle
  Accès répétés à une même adresse



Le principe de localité

- Localité spatiale
  - Accès à des adresses consécutives
  - Le coût d'un accès bloc est beaucoup moindre que le coût de n accès indépendants : T(bloc) = Ts + n Ti



Conséquence
Transferts blocs d'adresses consécutives
Unité Cache : *Ligne*
Unité MV : *page*

Le principe de localité

- Localité spatiale
  - Accès à des adresses consécutives
  - Le coût d'un accès bloc est beaucoup moindre que le coût de n accès indépendants : T(bloc) = Ts + n Ti
- Conséquence

Transferts blocs d'adresses consécutives

  - Unité Cache : *Ligne*
  - Unité MV : *page*

Le produit Matrice-Vecteur

Le produit Matrice-Matrice

Conséquences pour le programmeur

- Les accès par indirection peuvent coûter TRES cher

```
X[T[i]] = …
= X[T[i]]
```

- Attention aux conventions des compilateurs

Conséquences pour le programmeur

Des gains de performances importants peuvent être obtenus en localisant les accès

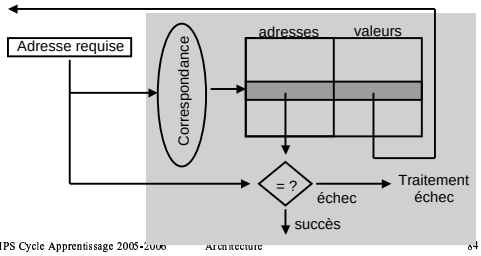
- Il existe des outils d'analyse des performances processeur et mémoire
  - VTune pour Windows
  - Associés à des compilateurs commerciaux sous Linux/Unixes
  - Quelques logiciels libres produits de recherche
- L'optimisation automatique est un sujet de recherche

Plan

- Motivations
- La Mémoire Centrale
- La Mémoire Virtuelle

Principe

Le cache est plus petit que la MP
=> plusieurs adresses MP sur un même emplacement cache



Principe

- L'accès au cache est effectué
  - Pour la lecture de TOUTES les instructions
  - Pour TOUS les accès mémoire

$$Ta = Th + e \times P$$

Ta = Temps d'accès moyen
Th = Temps accès cache
E = taux d'échec
P = Pénalité d'échec

Traitement des échecs

- En lecture
  - Principe de localité : la ligne absente est chargée dans le cache
  - Elle remplace la ligne occupant précédent : conflit

adresses	valeurs
00001000	12
00001001	34
00001002	56
00001003	78

Traitement des échecs

- En lecture
  - Principe de localité : la ligne absente est chargée dans le cache
  - Elle remplace la ligne occupant précédent

Adresse requise
00002002

adresses	valeurs
00002000	AB
00002001	CD
00002002	EF
00002003	01

Traitement des écritures - Succès

- Write through : écriture cache et MP
- Write back : écriture cache
  - Problème de cohérence : la MP n'est plus identique au cache
  - Marquage de l'écriture : dirty bit
  - A l'éviction de la ligne, recopie de la ligne modifiée en MP

+ Evite des accès MP en cas d'écritures multiples

- Double la pénalité d'échec en lecture

Traitement des écritures - Echec

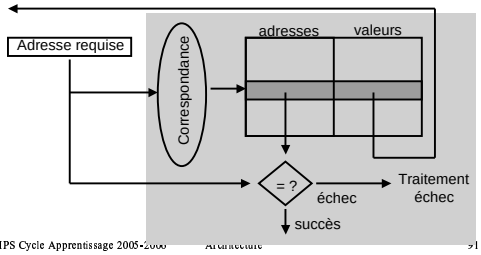
- Ecriture allouée (write allocate)
  - Si échec, chargement de la ligne correspondante et écriture
- Ecriture non allouée (no write allocate)
  - Si échec, écriture uniquement dans la mémoire principale
- NB : Les 4 combinaisons {WT ; WB}x{WA ; NWA} sont possibles

Cohérence

- La MP peut être modifiée, par exemple par E/S
- Les microprocesseurs intègrent le matériel nécessaire pour être notifiés sur leur caches interne des évènements extérieurs
- Chipsets pour les caches externes

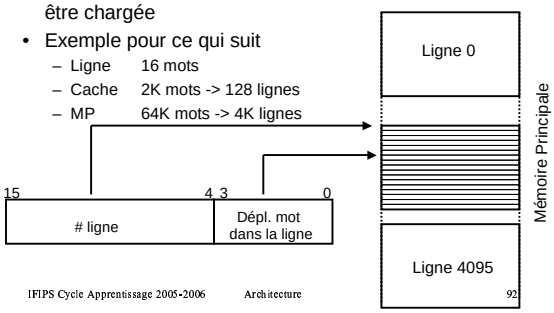
Principe

Le cache est plus petit que la MP
=> plusieurs adresses MP sur un même emplacement cache



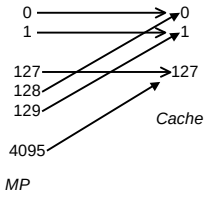
Correspondance

- Définit les lignes de cache où une ligne mémoire peut être chargée
- Exemple pour ce qui suit
  - Ligne 16 mots
  - Cache 2K mots -> 128 lignes
  - MP 64K mots -> 4K lignes

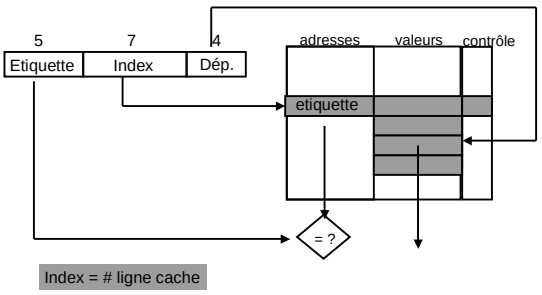


Correspondance directe

- Un seul emplacement possible
  # ligne cache = # ligne mémoire modulo taille cache



Correspondance directe

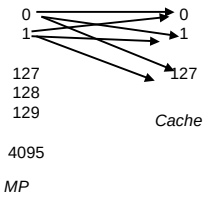


Correspondance directe

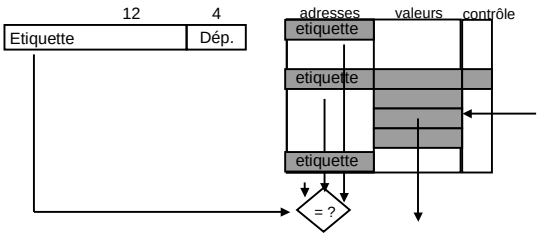
- Temps de comparaison négligeable
- Sous-exploitation du cache : probabilité de conflit élevée

Correspondance complètement associative

- Tous les emplacements sont possibles



Correspondance complètement associative



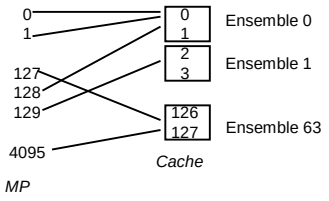
Correspondance complètement associative

- Temps de comparaison prohibitif
- Pas utilisé pour les caches instructions et données
- Utilisé pour des caches de petite taille, pour le fonctionnement interne du processeur ex. TLB

Correspondance associative par ensemble

- Un ensemble = quelques lignes de cache (2 ou 4)

  # ensemble cache = # ligne mémoire modulo nombre d'ensembles



Correspondance associative par ensemble

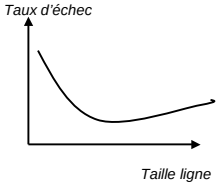
- Politique de remplacement
- Quelle est la ligne remplacée dans un ensemble si conflit ?
- LRU = Least Recently Used
- Implémenté en pseudo-LRU si plus de 2 lignes par ensemble

Améliorer la performance

- Diminuer le taux d'échec
  - Programmation
  - Matériel
- Diminuer la pénalité d'échec
  - Matériel : organisation mémoire DRAM améliorées

Taux d'échec et organisation cache

- Origine des échecs
  - Démarrage à froid
  - Capacité
  - Conflit



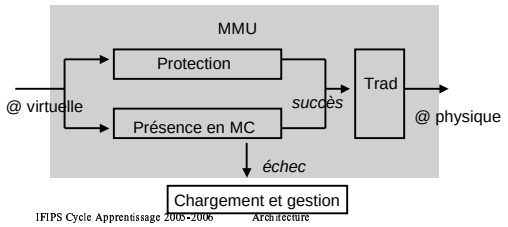
Augmenter taille cache : technologie

Augmenter taille ligne
- Diminue échecs de démarrage
- Augmente les conflits

Degré d'associativité
- Diminue les conflits
- Technologie
- Règle « 2:1 »

Mémoire virtuelle

- Les programmes ne connaissent que les adresses virtuelles
- La traduction est sur le chemin critique



Gestion

- Accès disque
  - Pagination : localité spatiale
  - Associativité totale
  - Gestion logicielle
  - Write Back

Traduction

- Table des pages
- Tableau de taille nombre de pages virtuelles
- Entrée : descripteur de l'adresse physique, soit en mémoire, soit sur disque plus contrôle
- Traduction = indexer la table des pages par l'adresse virtuelle

TLB

- La table des pages est une grosse structure, en MC au mieux
- Cache de la table des pages : TLB
- C'est le TLB qui est sur le chemin critique : un défaut de TLB entraîne le rechargement de l'entrée correspondante et le redémarrage de la traduction via le TLB
- Traitement des défauts de TLB
  - Matériel : rapide, fige l'OS
  - Logiciel très bas niveau

Pour en savoir plus…

http://www.lri.fr/ENSEIGNANTS/LM/archi/L4.html