Travaux Pratiques Archi SLE-3A
Cours de Frédéric Pétrot

Implantations et évaluations de quelques prédicteurs de branchements

Durée : 3 heures

1 Organisation

Ce TP se fait soit seul, soit en binôme, à votre convenance. Le travail demandé TP fait l'objet d'un rendu contenant un court rapport qui inclut les *résultats* d'expérimentations et un *tarball* (ou un git) contenant vos sources à fournir en fin de séance. Un gabarit en LaTeX qui reprend le predicteur BHT avec un compteur à saturation est donné en exemple. Vous avez également un script fourni pour vous permettre de systématiser les simulations et l'obtention des graphes des résultats. Ce script marche avec le prédicteur BHT qui a deux arguments, il est donc nécessaire de le modifier pour l'utiliser avec un nombre d'arguments différent.

2 Introduction

On cherche à étudier le comportement en situation réelle de différents prédicteurs de branchement. Pour cela, on va développer un modèle en C++ de chacun de ces prédicteurs sur lequel on fera passer un ensemble de traces d'exécutions et qui fournira en sortie le ratio de mauvaises prédictions (en fait le nombre de *Miss Prediction per Kilo Instructions*, MPKI). On fera varier les différents paramètres du prédicteur, en particulier la taille des tables, afin d'en extraire un comportement asymptotique (par benchmark), dont on tirera un graphe.

L'infrastructure dans laquelle insérer le modèle de prédicteur est celle qui a été utilisée par le *The 5th JILP Championship Branch Prediction Competition (CBP-5)*[1]. Cette infrastructure effectue la lecture des traces et l'appel à des fonctions qui implantent le prédicteur, et donne en sortie diverses statistiques dont la valeur du MPKI.

Le prédicteur par défaut dans l'archive que je vous fournis est un prédicteur à saturation « n-bits » vu en cours et rappelé figure 1 en considérant un prédicteur bimodal. Dans une case donnée, un branchement « prédit pris » aura une valeur $v \geq 2^{n-1}$ et « prédit non pris » aura une valeur $v < 2^{n-1}$. Nous ferons en séance une analyse rapide du code afin de comprendre comment il a été implanté dans l'infrastructure.

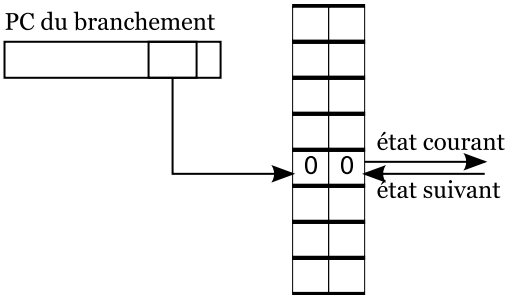


Figure 1 – Prédicteur bimodal

3 Travail demandé

On implantera différents prédicteurs en prenant soin de paramétrer les tailles afin de pouvoir lancer facilement plusieurs exécutions (*cf.* l'exemple fourni) et ainsi pouvoir tracer des courbes et voir les asymptotes.

Les prédicteurs que l'on pourra d'implanter sont les suivants :

1. un prédicteur global simple avec un historique des branchement de longueur (nombre de bits) H permettant d'atteindre une entrée bimodale utilisant le graphe d'état n°1 (la *Pattern History Table*). L'historique des branche-

1. Dont on trouvera la version originale sur `https://www.jilp.org/cbp2016/framework.html`

ments est un registre à décalage à gauche dans lequel on injecte sur le poids faible la décision prise ($H = 3$ dans la figure 2)[2] ;

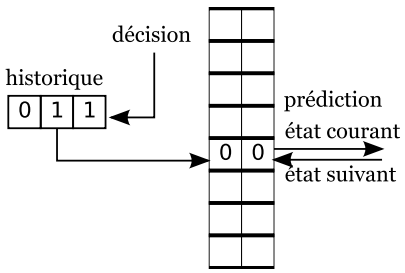


FIGURE 2 – Prédicteur global simple

2. un prédicteur corrélé qui utilise un historique sur H bits permettant de sélectionner une PHT parmi 2^H et utilisant l'adresse du branchement pour indicer cette table, comme illustré sur la figure 3 ;

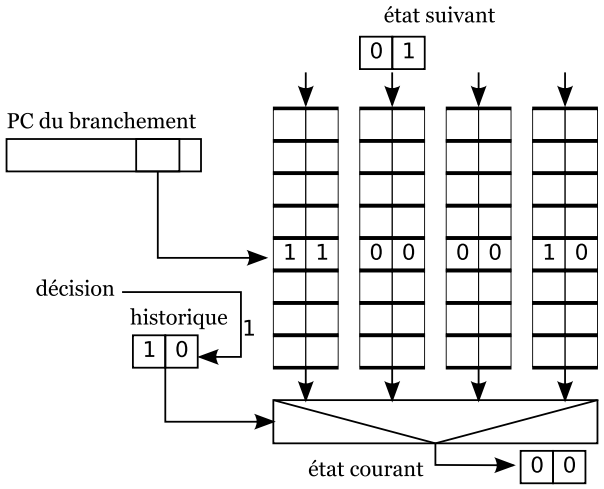


FIGURE 3 – Prédicteur corrélé

3. un prédicteur local qui utilise une table d'historiques sur H bits indexée par les poids faibles de l'adresse du branchement. En utilisant l'entrée d'historique, on indice une PHT bimodale de taille 2^H, comme illustré sur la figure 4 ;

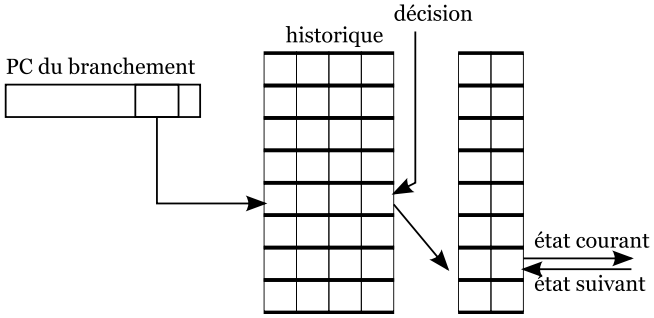


FIGURE 4 – Prédicteur local

4. et pour finir un prédicteur mixte (celui de l'alpha 21264) qui utilise un prédicteur global simple avec un historique de $H_g = 12$ (soit 4096 entrées), et un prédicteur local comme celui de la précédente question, avec $H_l = 10$ par

2. Ce pourrait tout aussi bien être un registre à décalage à droite avec injection sur les poids forts, comme vu en cours.

entrée de la table d'historique (soit 1024 entrées). La PHT indexée est un simple prédicteur de type bimodale, mais sur 3 bits, implantée avec un compteur à saturation.

Le choix du prédicteur à utiliser se fait grâce à une BHT de 4K entrées de 2 bits. Le compteur est incrémenté lorsque le prédicteur *prédit* est correcte et l'autre prédicteur a fait le mauvais choix, et est décrémenté dans le cas opposé (*cf.* la machine d'état de la figure 5).

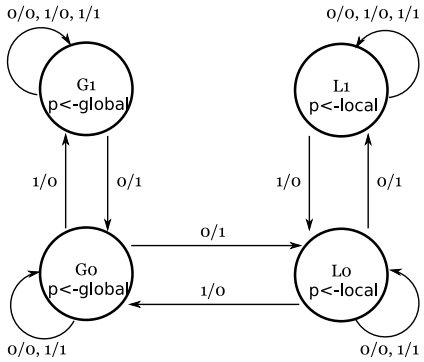


FIGURE 5 – Machine à états du choix du prédicteur. Légende : $global/local$, 0 décision incorrecte, 1 décision correcte

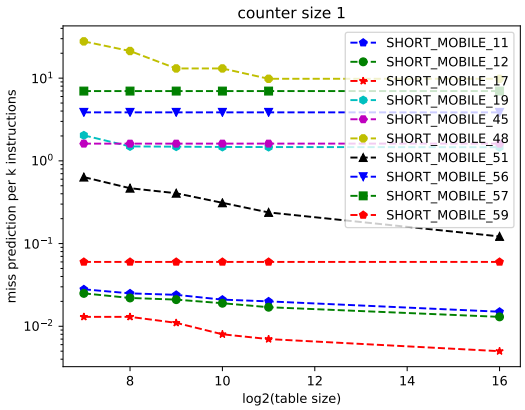
On ne tentera pas (du moins dans le temps imparti) de faire une version paramétrable de ce dernier prédicteur.

Il y a un script `seoc3a-doit.sh` dans le répertoire `script` qui lance automatiquement l'exécution sur un sous-ensemble des traces pour un prédicteur donné, qu'il a fallut préalablement compiler dans le répertoire `sim`. Il y a dans le script pour le prédicteur n-modal 2 boucles imbriquées : la boucle externe, indice i, fait varier le nombre de bits du compteur, et la boucle interne, d'indice j, fait varier le nombre de bits de PC à utiliser pour indexer les tables. Vous serez amené à modifier les valeurs des bornes en fonction des prédicteurs, voir à changer un peu cette partie si vous faite des prédicteurs exotiques.

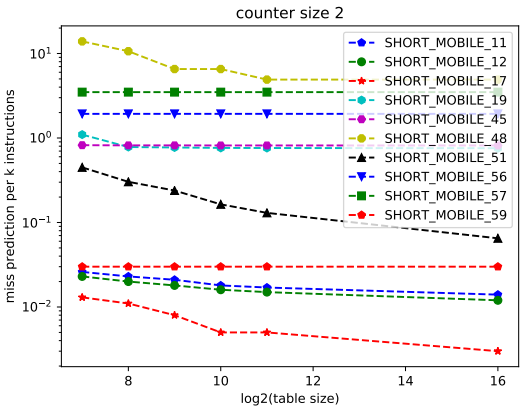
Le script `seoc3a-doit.sh` produit ses résultats dans `../results/xxx`, ou `xxx` est un répertoire dont le nom peut-être choisi à l'envie. Il les analyse ensuite et génère des programmes `python` qui servent à tracer des courbes (avec `matplotlib`). Ce sont ces courbes que l'on vous demande d'analyser.

On donne sur la figure 6 le type de résultat attendu pour le prédicteur 1 bit.

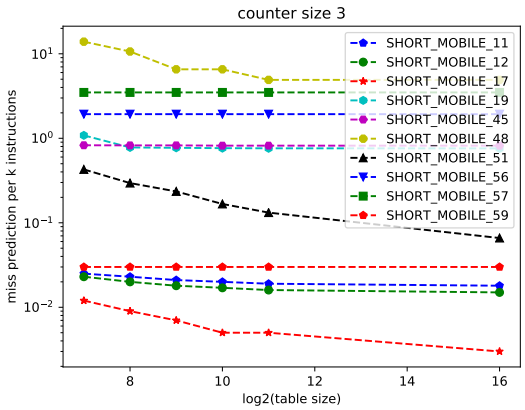
Le sous-répetoire `seoc3a/rendu` contient un exemple de .tex permettant de faire un petit rapport sur le travail.



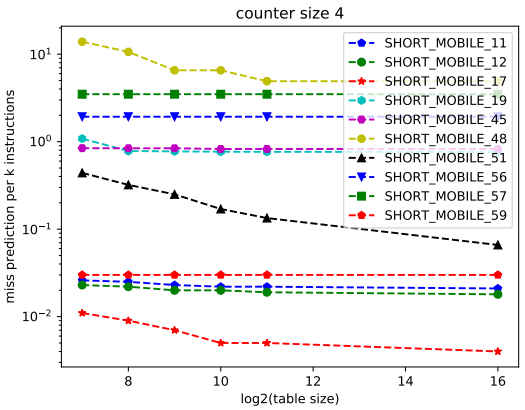
(a) Table d'historique 1 bit



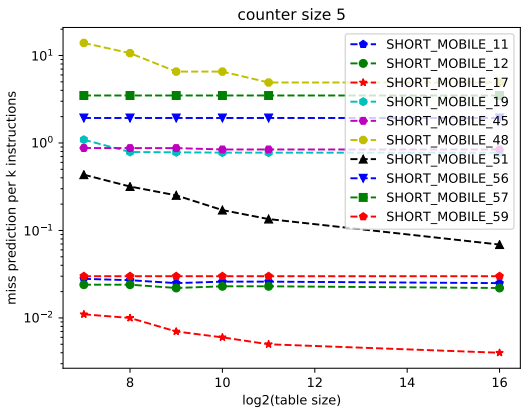
(b) Table d'historique 2 bits



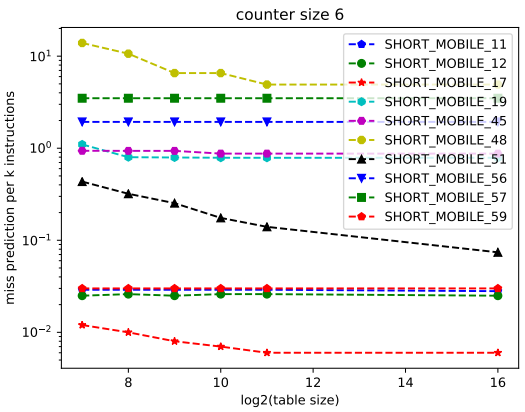
(c) Table d'historique 3 bits



(d) Table d'historique 4 bits



(e) Table d'historique 5 bits



(f) Table d'historique 6 bits

FIGURE 6 – Les courbes de MPKI