Architecture des processeurs et optimisation
Novembre 2010 **Documents autorisés - Durée 2h**

utiliser impérativement les feuilles quadrillées fournies pour les schémas simplifiés

L'objet de cette étude est de comparer la performance de deux réalisations de Mips-32.

La première est la réalisation classique sur un pipeline de 5 étages présenté en cours. Cette réalisation est appelée **Mips**.

La seconde, appelée **P9**, est un pipeline à 9 étages : `IF1`, `IF2`, `IF3`, `DEC`, `EXE`, `MM1`, `MM2`, `MM3` et `WBK`. Dans cette réalisation, les accès à la mémoire s'effectuent en trois cycles. Au début de l'étage `IF1`, l'adresse de l'instruction est envoyée à la mémoire et l'instruction correspondante est récupérée à la fin du cycle `IF3`. De la même manière, l'accès à la mémoire de données se déroule en trois cycles. Dans l'étage `DEC`, l'instruction est décodée et les opérandes sont préparés. Le **calcul de l'adresse** de l'instruction suivante s'effectue dans l'étage `EXE`. Dans l'étage `WBK`, le résultat est écrit dans le banc de registres. Dans cette réalisation, il n'y a pas de bypass vers l'étage `MM1`.

1- Quels sont les impacts de la réalisation **P9** sur le compilateur.

2- Dans la réalisation Mips il y a des dépendances de données d'ordre 1, 2 et 3 et 8 bypass. Quelles sont les dépendances de données dans P9. Quels sont les bypass dans P9.

L'objectif de cet exercice est de comparer la performance d'exécution de 2 fonctions dans les deux réalisations.

Dans une machine, les nombres entiers sont représentés sur 1, 2 ou 4 octets. Mais, dans certains cas on peut souhaiter représenter des entiers sur un plus grand nombre d'octets (par exemple, lorsque l'on souhaite manipuler la mentisse des nombres réels avec une très grande précision). Dans ce cas, l'entier est représenté par un tableau d'octets. L'octet de poids faible du nombre est placé à l'index 0 du tableau (représentation *Little Endian*). Dans cet exercice, on ne considère que les entiers naturels.

Exercice I :

La fonction `Cmp` compare deux nombres entiers dans cette représentation. Elle renvoie une valeur négative si le premier nombre est strictement inférieur au second, la valeur 0 si les deux nombres sont identiques ou une valeur positive si le premier nombre est strictement supérieur au second. Les deux nombres sont codés sur deux tableaux de même taille.

```
int Cmp (unsigned char *n1, unsigned char *n2, unsigned int size)
  {
  unsigned int i = 0;

  for (i=size ; i>0 ; i--)
    {
    if (n1 [i-1] != n2 [i-1])
      break;
    }
  return (n1 [i-1] - n2 [i-1]);
  }
```

Le code de la boucle principale de `Cmp` en assembleur Mips (non pipeline) est le suivant. On suppose qu'à l'entrée de la boucle le registre R8 pointe sur la dernière case du tableau `n1`, le registre R9 pointe sur la dernière case du tableau `n2` et que R4 contient l'adresse du tableau `n1`.

```
_Loop :          Lbu     r10, 0 (r8 )
                 Lbu     r11, 0 (r9 )
                 Bne     r10, r11, _EndLoop
                 Addiu   r9 , r9 , -1
                 Addiu   r8 , r8 , -1
                 Bne     r4 , r8 , _Loop

_EndLoop :       ...
```

3- Modifier le code de la boucle de `Cmp` pour qu'il soit exécutable sur la réalisation **Mips**. Analyser l'exécution de cette boucle à l'aide d'un schéma simplifié. Calculer le nombre moyen de cycles par itération. Calculer le CPI et le CPI-utile.

4- Modifier le code de la boucle de `Cmp` pour qu'il soit exécutable sur la réalisation **P9**. Analyser l'exécution de cette boucle à l'aide d'un schéma simplifié. Calculer le nombre moyen de cycles par itération. Calculer le CPI et le CPI-utile.

5- Quel rapport de fréquences d'horloge entre **P9** et **Mips** doit-on observer pour que l'exécution de ce code soit plus efficace dans **P9** que dans **Mips**.

6- Pour la réalisation **Mips**, modifier l'ordre des instructions de `Cmp` afin d'éviter au maximum les cycles perdus. Il n'est pas nécessaire de faire un schéma mais d'indiquer simplement les cycles de gel. Calculer le nombre moyen de cycles par itération, le CPI et le CPI-utile.

7- Même question pour **P9.**

8- Pour la réalisation **Mips**, optimiser le code de la boucle de `Cmp` à l'aide de la technique *Software pipeline* en indiquant clairement le nombre d'étages et les instructions contenus dans chaque étage. Il n'est pas nécessaire de faire un schéma mais d'indiquer simplement les cycles de gel. Calculer le nombre moyen de cycles par itération, le CPI et le CPI-utile.

9- Pour la réalisation **P9**, optimiser le code de la boucle de `Cmp` à l'aide de la technique *Software pipeline* en indiquant clairement le nombre d'étages et les instructions contenus dans chaque étage. Analyser le code obtenu à l'aide d'un schéma simplifié. Calculer le nombre moyen de cycles par itération, le CPI et le CPI-utile.

10- Pour la réalisation **Mips**, dérouler une fois le code de la boucle de `Cmp` (traitement de 2 éléments à chaque itération) et optimiser. Il n'est pas nécessaire de faire un schéma mais d'indiquer simplement les cycles de gel. Calculer le nombre moyen de cycles par itération, le CPI et le CPI-utile.

11- Même question pour P9.

Exercice II :

La fonction `Srl` réalise un décalage à droite logique d'un nombre entier codé sur un tableau. Le nombre de décalage est un nombre d'octets.

```
unsigned char *Srl (unsigned char *n, unsigned int size, unsigned int sham)
         {
         unsigned int i;

         for (i=sham ; i<size ; i++)
           n [i-sham] = n [i] ;

         return (n);
         }
```

Le code de la boucle principale de `Srl` en assembleur Mips (non pipeline) est le suivant. On suppose qu'à l'entrée de la boucle :

- R4 pointe sur la première case du tableau `n`
- R8 pointe sur la case d'index `sham` du tableau `n`
- R9 contient l'adresse de fin du tableau `n`

```
_Loop :          Lbu     r12, 0 (r8 )
                 Sb      r12, 0 (r4 )
                 Addiu   r4 , r4 , 1
                 Addiu   r8 , r8 , 1
                 Sltu    r10, r8 , r9
                 Bne     r10, r0 , _Loop
```

12- Modifier le code de la boucle de `Srl` pour qu'il soit exécutable sur la réalisation **Mips**. Il n'est pas demandé de faire un schéma mais d'indiquer simplement les cycles de gel. Calculer le nombre moyen de cycles par itération. Calculer le CPI et le CPI-utile.

13- Même question pour P9.

14- Pour la réalisation **Mips**, optimiser le code de la boucle de `Srl` à l'aide de la technique *Software pipeline* en indiquant clairement le nombre d'étages et les instructions contenus dans chaque étage. Il n'est pas demandé de faire un schéma mais d'indiquer simplement les cycles de gel. Calculer le nombre moyen de cycles par itération, le CPI et le CPI-utile.

15- Même question pour P9.