Magistère de Mathématiques Fondamentales & Appliquées et d'Informatique.

Corrigé du contrôle final du cours 1999 :
Théorie de des Technologies de l'Information.

L'objet du problème est d'étudier des algorithmes, logiciels et matériels, pour transposer les matrices. La transposée d'une matrice carrée $n \times n$ $B = [b[i,j] \ : \ 0 \leq i,j < n]$ est la matrice $tr(B) = \tilde{B} = [b[j,i] \ : \ 0 \leq i,j < n]$.

On s'intéresse ici aux matrices binaires (soit $b[i,j] \in \{0,1\}$) dont la taille est une puissance de deux : $n = 2^l$, et $n \times n = 4^l$. Une telle matrice se représente par n mots consécutifs $M[0..n-1]$, dans une mémoire M dont le nombre b de bits par mot mémoire est au moins égal à n : $b \geq n$. Le contenu de la case mémoire $M[j]$, d'adresse j pour $0 \leq j < n$, est le mot binaire $M[j] = b_{0,j} \cdots b_{i,j} \cdots b_{n-1,j}$ qui représente la ligne j de la matrice B, avec $b_{i,j} = b[i,j]$ pour $i \leq n$, et $b_{i,j} = 0$ pour $i > n$. Le problème à résoudre : étant donnée une matrice B, présentée en mémoire comme indiqué, calculer la matrice transposée $T = \tilde{B}$, et la ranger sous le même format.

1 Transposition de matrice par logiciel

Le fragment de code $tr2x2$ qui suit résoud le problème pour une matrice 2×2 :

$$\begin{array}{lcrcl} T[0] & = & (B[0] \,\&\, 1) & | & ((B[1] \,\&\, 1) << 1); \\ T[1] & = & ((B[0] >> 1) \,\&\, 1) & | & (B[1] \,\&\, 2); \end{array}$$

Pour ceux qui ne lisent, ni C, ni Java, ce code peut s'écrire :

$$\begin{array}{lcl} T[0] & = & (B[0] \cap 1) \cup 2 \times (B[1] \cap 1), \\ T[1] & = & ((B[0] \div 2) \cap 1) \cup (B[1] \cap 2). \end{array}$$

Le *mot machine* du processeur sur lequel un tel code va s'exécuter a 64 bits. Le processeur comprend une mémoire $M[0..]$, et quatre registres $R[0], R[1], R[2]$ et $R[3]$. Le jeu des instructions permet de calculer, chacune en un cycle, les *opérations machine* suivantes :

- échanges entre mémoire et registre : $M[a] := R[r]$ et $R[r] := M[a]$, pour toute adresse mémoire $a \geq 0$, et numéro de registre $0 \leq r < 4$;
- opérations entre registres, $R[d] := R[s]$ op O, où l'opérande O représente, soit le contenu d'un registre $O = R[r]$ pour $0 \leq r < 4$, soit une constante sur 64 bits, et l'opération op est à choisir parmi : $\cap$, $\cup$, $\oplus$, $+$ et $-$;
- décalages binaires, à droite $R[d] := R[s] >> O$ (soit $R[d] = R[s] \div 2^O$), comme à gauche $R[d] := R[s] << O$ (soit $R[d] = R[s] \times 2^O$), pour O entier.

Question 1 *1. Ecrire une suite d'instructions machines pour réaliser le code $tr2x2$ - elle ressemble à ce qu'un compilateur produirait automatiquement. Compter les cycles nécessaires à son exécution.*

2. *Ecrire un code* $tr2x2x32$ *qui transpose 32 matrices* 2×2*, à la fois : les sorties correspondant aux deux mots d'entrées*

   $B[0] = x_0x_1x_2x_3 \cdots x_{62}x_{63}$ *et*

   $B[1] = x'_0x'_1x'_2x'_3 \cdots x'_{62}x'_{63}$*, sont les deux mots de 64 bits :*

   $T[0] = x_0x'_0x_2x'_2 \cdots x_{62}x'_{62}$ *et*

   $T[1] = x_1x'_1x_3x'_3 \cdots x_{63}x'_{63}$.

   Compiler $tr2x2x32$*, et compter les cycles nécessaires à son exécution sur notre processeur. Note : il existe une solution pour laquelle ce nombre est identique à celui trouvé pour la question précédante.*

Réponse 1 *1. Le code* $tr2x2$ *se compile en 12 instructions : une pour chacun des 8 opérateurs dans la formule, et 4 cycles de transfert avec la mémoire. Remplacer, dans la suite d'instructions qui suit :* $c1 = 1$, $c2 = 2$, $s = 1$ *soit* $CT(1, 2, 1)$.

```
                                    // get the operands from memory
R[0]: =M[0];                        // R[0] = a_0 a_1 ...
R[1]: =M[1];                        // R[1] = b_0 b_1 ...

                                    // replace, in the following code, c1=1, c2=2, s=1
                                    // comments only apply to the case c1=1, c2=2, s=1

                                    // macro code for CT2(c1, c2, s)
R[2]: =R[0] & c1;                   // R[2] = a_0 0 ...
R[0]: =R[0] >> s;                   // R[0] = a_1 ...
R[0]: =R[0] & c1;                   // R[0] = a_1 0 ...
R[3]: =R[1] & c1;                   // R[2] = b_0 0 ...
R[3]: =R[3] << s;                   // R[3] = 0 b_0 ...
R[1]: =R[1] & c2;                   // R[1] = 0 b_1 ...
R[1]: =R[0] | R[1];                 // R[1] = a_1 b_1 ...
R[2]: =R[2] | R[3];                 // R[2] = a_0 b_0 ...
                                    // end of macro CT2

                                    // store the results, back in memory
M[0]: =R[2];
M[1]: =R[1];
                                    // end of first CT2
```

2. *On généralise le code $tr2x2$ en introduisant 3 paramètres entiers, $c1$, $c2$ et s :*

```
T0 = (B0 & c1) | (B1 & c1) << s;
T1 = ((B0 >> s) & c1) | (B1 & c2);
```

Il suffit alors d'utiliser ce code avec les constantes : s=1, $c1 = \sum_{k<32} 4^n = (4^{32}-1)/3$ et $c2 = 2 \times c1 = \neg c1 \pmod{2^{64}}$. La représentation binaire de $c1$ est donnée par les 64 premiers bits du nombre périodique $-1/3 = (10)$, celle de $c2$ par (01). Comme la présentation donnée du code machine comprend déjà les paramètres, il faut choisir ces valeurs pour les constantes. Le nombre de cycles, nécessaires pour l'exécuter, vaut 12.

Question 2 *Ecrire (dans le style de $tr2x2$) un code $tr4x4$ pour transposer les matrices binaires 4×4. Le compiler, et compter les cycles nécessaires à son exécution.*

Réponse 2 *Le code (Jazz) qui suit reprend la definition de $CT2$, et il donne celle de $tr4x4$.*

```
CT2(B0, B1, c1, c2, s)=(T0, T1)
where
 T0 = (B0 & c1) | shiftL(B1 & c1, s);
 T1 = (shiftR(B0, s) & c1) | (B1 & c2);
end where;
shiftR(x, s)=x div 2**s;                    // shift right:  x >> s
shiftL(x, s)=x * 2**s;                      // shift left:   x << s

CTM4(B:_[4])=C:_[4]
where
 c1=#1010; c3=#1100;
 c2=#0101; c4=#0011;
 (C[0],C[1])= CT2(B[0], B[1], c1, c2, 1);
 (C[2],C[3])= CT2(B[2], B[3], c1, c2, 1);
 (T[0],T[2])= CT2(C[0], C[2], c3, c4, 2);
 (T[1],T[3])= CT2(C[1], C[3], c3, c4, 2);
end where;
```

Une fois compilé comme suit, ce code s'exécute en 48=12×4 cycles.

```
                                        // get the operands for first CT2
 R[0]: =M[0];
 R[1]: =M[1];
                          // macro code for CT2(c1=#1010, c2=#0101, s=1)
                                                    // 8 instructions
                                                    // store the results
```

```
M[0]: =R[2];
M[1]: =R[1];
                                        // end of first CT2

                                        // get the operands for second CT2
R[0]: =M[1];
R[1]: =M[2];
                                // macro code for CT2(c1=#1010, c2=#0101, s=1)
                                        // 8 instructions
                                        // store the results
M[1]: =R[2];
M[2]: =R[1];
                                        // end of second CT2

                                        // get the operands for third CT2
R[0]: =M[0];
R[1]: =M[2];
                                // macro code for CT2(c1=#1010, c2=#0101, s=1)
                                        // 8 instructions
                                        // store the results
M[0]: =R[2];
M[2]: =R[1];
                                        // end of third CT2

                                        // get the operands for last CT2
R[0]: =M[1];
R[1]: =M[3];
                                // macro code for CT2(c1=#1010, c2=#0101, s=1)
                                        // 8 instructions
                                        // store the results
M[1]: =R[2];
M[3]: =R[1];
                                        // end of last CT2
```

Question 3 *Décrire la structure d'un code $tr64x64$ pour transposer les matrices binaires 64×64. Analyser le nombre de cycles nécessaires à son exécution.*

Montrer qu'il existe une solution, pour transposer les matrices $n \times n$, avec un nombre d'opérations proportionnel à $Cnlog(n)$. Evaluer C.

Réponse 3 *Si on décompose la matrice B en quatre sous blocs de tailles égales, la transposée $\tilde{B}$ se forme à partir des transposées des quatre blocs, par :*

$$B = \begin{bmatrix} B_{0,0} & B_{0,1} \\ B_{1,0} & B_{1,1} \end{bmatrix} \qquad \tilde{B} = \begin{bmatrix} \tilde{B_{0,0}} & \tilde{B_{1,0}} \\ \tilde{B_{0,1}} & \tilde{B_{1,1}} \end{bmatrix}$$

En généralisant à partir de la solution de la question 1.2, on voit que : un code pour transposer une matrice $n \times n$ en T cycles peut être modifié, de façon à transposer en parallèle, $p = b/n$ matrices $n \times n$ en T cycles, où b est la largeur du mot machine.

En combinant ces deux observations, on arrive à un algorithme pour transposer les matrices $n \times n$ en $nlog(n)$ opérations, du moins tant que $n \leq b$, avec b le nombre de bits par mots.

On transpose une matrice $n \times n$ $B = \begin{bmatrix} B_{0,0} & B_{0,1} \\ B_{1,0} & B_{1,1} \end{bmatrix}$, sur une machine n bits, comme suit.

1. *Transposer, récursivement et en parallèle, les deux sous matrices $n/2 \times n/2$: $\tilde{B}0 = \begin{bmatrix} \tilde{B_{0,0}} & \tilde{B_{0,1}} \end{bmatrix}$.*
2. *Transposer, récursivement et en parallèle, les deux sous matrices : $\tilde{B}1 = \begin{bmatrix} \tilde{B_{1,0}} & \tilde{B_{1,1}} \end{bmatrix}$.*
3. *Echanger $\tilde{B_{0,1}}$ avec $\tilde{B_{1,0}}$, pour former $\tilde{B}$.*

Le nombre de cycles nécessaires pour échanger les deux sous matrices (étape 3) est $6n$, au plus. En effet, pour chacunne des $n/2$ paires de mots de n bits du résultat, on calcule les 12 cycles du code $CT2$.

Le nombre T_n de cycles pour transposer une matrice $n \times n$ par cette méthode est donc donné par : $T_1 = 0$, et $T_n = 2T_{n/2} + 6n$, soit $T_n = 6nlog_2(n)$. Pour $n = 64$, on trouve $T_{64} = 6 \times 64 \times 6 = 2304$ cycles.

Le code qui suit donne, en Jazz, une version récursive de cet algorithme.

```
CTM(n: int)(B: _[n])=T: _[n]
where                                  // Corner Turn Memory: recursive version
 if n==1 then T=B else
  n' = n div 2;
  Y0 = CTM(n')(B[0..n'-1]);
  Y1 = CTM(n')(B[n'..n-1]);
  (c1,c2)= ((2**n'-1)/(1-2**n), ~c1);
  for k¡n' do
   (T[k], T[k+n'])=CT2(Y0[k], Y1[k], c1, c2, n');
  end for;
 end if;
end where;
```

Le code qui suit donne une version itérative du même algorithme.

```
CTMf(n: int)(B: _[n])=Y: _[n]
where                                  // Corner Turn Memory: iterative version
 assert(n==2**l);                                                  // n==2**l
```

```
 l=prefixLength(n)-1;
 Y=V[l];                                        // where to find the result
 V[0]=B;                                        // initial array value
 p[l]=n;                                        // p[l]=2**l
 for v¡l do
  p[v]=2**v;
  p2[v]=2**p[v];
  c[v]=-p2[v]/(1+p2[v]);                        // c=[(01),(0011),(00001111), ...]
  b[v]= n div p[v+1];                           // shift amount
  for m¡b[v] do
   for k¡p[v] do
     with i=m*p[v+1]+k; i'=i+p[v] do
       (V[v+1][i],V[v+1][i'])
       =CT2(V[v][i], V[v][i'],~c[v],c[v], p[v])
     end with;
   end for;
  end for;
 end for;
end where;
```

2 Transposition de matrice par matériel

Question 4 *Réaliser un circuit digital synchrone pour transposer les matrices 2×2 : il possède deux bits d'entrée $B[0], B[1]$, et deux bits de sortie $T[0], T[1]$. Si*

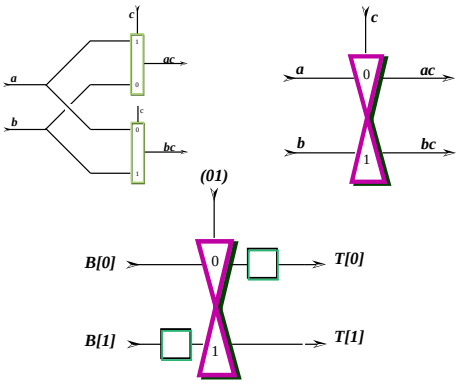
$B[0] = x_0 x_1 x_2 x_3 \cdots x_{2^N} x_{2^N+1} \cdots$ *et*

$B[1] = x'_0 x'_1 x'_2 x'_3 \cdots x'_{2^N} x'_{2^N+1} \cdots$ *representent les suites (infinies) des bits d'entrée à chaque cycle* $N \in \mathbf{N}$*, les valeurs des bits de sortie doivent être, à un retard près (que l'on doit expliquer et justifier) :*

$T[0] = x_0 x'_0 x_2 x'_2 \cdots x_{2^N} x'_{2^N} \cdots$ *et*

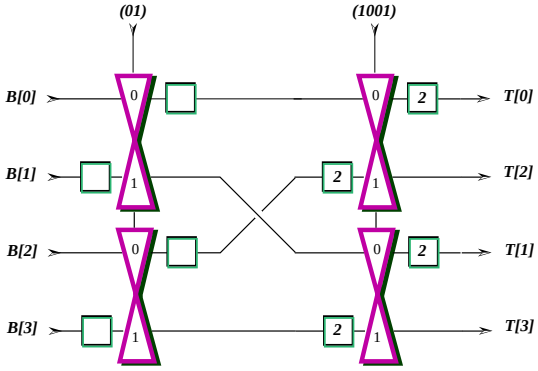
$T[1] = x_1 x'_1 x_3 x'_3 \cdots x_{2^N+1} x'_{2^N+1} \cdots$.

Réponse 4 *Commme le bit 0 de $T[1]$, qui est égal au bit 1 de $B[0]$, ne rentre dans le circuit qu'au cycle 2, la délai minimal pour transposer les matrices 2×2 est de un cycle. Le circuit suivant calcule la fonction de $tr2x2$, à ce retard de 1 cycle près.*



Question 5 *Réaliser un circuit digital synchrone pour transposer les matrices 4×4 : il possède quatre bits d'entrée $B[0..3]$, et de sortie $T[0..3]$. Analyser sa taille (nombre de registres et de portes logiques) et le délai nécessaire avant l'apparition de sorties significatives.*

Réponse 5 *Le circuit suivant reprend l'algorithme $tr4x4$.*

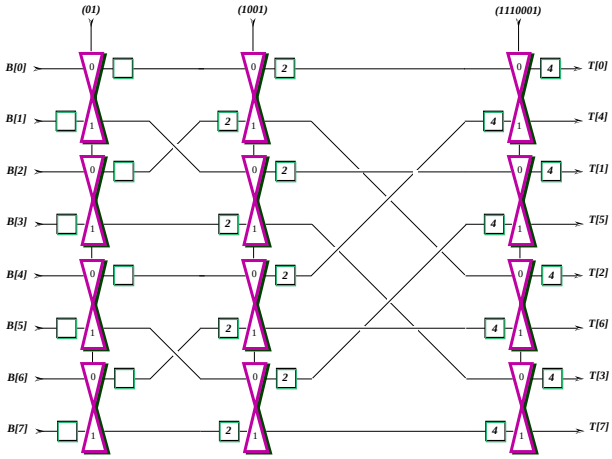


Les signaux de contrôle sont les sorties d'un compteur 2 bits, avec retard d'un cycle sur le second bit. Les numéros dans les registres indiquent la profondeur du registre à décalage (ici 2 bits).

Avant que les bits significatifs n'apparaissent en sortie, le retard est maintenant de 3 cycles.

Question 6 *Concevoir un circuit pour tr ansposer les matrices $n \times n$: il possède n bits d'entrée $B[0..n-1]$, et de sortie $T[0..n-1]$. Analyser sa taille (nombre de registres et de portes logiques) et le délai nécessaire avant l'apparition de sorties significatives.*

Réponse 6 *En partant de l'agorithme récursif exposé à la question 3, on obtient un circuit en n couches, dessiné ici pour $n = 4$.*



Dans le cas général, ce circuit comporte $2nlog_2(n)$ *multiplexeurs, et* $n(n-1)$ *registres. Le délai en sortie est de* $n-1$ *cycles.*

Une solution plus classique est la mémoire à virage (Corner Turn Memory, dont on trouve le schéma ci-dessous, pour $n=4$*. Une différence par rapport au circuit récursif est que, le chargement alterne maintenant avec la lecture, au lieu de se faire en continu.*