Annexe B
Solutions des exercices

Chapitre 3

3.1.4.1 152=10011000 127=1111111 256=100000000 90=1011010 33=100001
65=1000001

3.1.4.2 1100110101=821 11101110111=1911
100010001000=2184 111111111111=4095
1010101010=682

3.1.4.3

```
  10111      110011      101010      10111001
+ 01011    + 011100    + 011111    + 00011001
  100010     1001111     1001001     11010010
```

3.1.4.4

```
  100110      111111      101110      110000
- 010110    - 101010    - 011111    - 010111
  010000      010101      001111      011001
```

3.1.4.5

```
  ABCD        lEFO        AAAA        CDEF
+ 4321      + 9042      + OAAA      + 1221
  EEEE        AF32        B554        E010
```

3.1.4.6

```
  34A2        F975        54EA        FFFF
- 0191      - 94AE      - lEFA      - EDCB
  3311        64C7        35F0        1234
```

3.8.1 Oui, c'est l'entier 3735596
Oui, ce sont les caractères ASCII étendu `0x39='9', 0x00=NUL, 0x2C=','.`
Oui, le premier mot est l'instruction `DECO` en mode d'adressage direct.

3.8.2 B001F4 qui comprend la constante 1F4 égale à 500 décimal. Instruction de comparaison.

3.8.3 L'édition de liens est l'établissement et l'ajustement des liens existant entre programme principal et divers sous-programmes venant d'autres fichiers ou bibliothèques.

Chapitre 4

4.2.1

```
a. 1234=4660     b. FADE=-1314     c. BABA=-17734
d. D2BC=-11588   e. 7AB3 =31411    f. 564E=22094
```

4.2.2

```
a. 1234=04D2     b. -1048=FBE8     c. 1492=05D4
d. 111=006F      e. -777=FCF7      f. 10=FFF6
```

4.2.3

```
a.   OA94    b.    B747   c.  2BC3  d.    6789    e.  0522   f.  CAFE
   + 6421       + CAFE     + 1234      + 9876      - 3502     - FADE
     6EB5         8245       3DF7        FFFF        D020       D020
```

4.2.4

```
a.    54321768    b.    00F9A920    c.    7FFF0123
    - 12345678        - 32145678        - 00012345
   (+ EDCBA988)      (+ CDEBA988)      (+ FFFEDCBB)
      41FDC0F0          CEE552A8          7FFDDDDE
```

4.5.1

```
LDA     X,d      Modifie l'accumulateur
STA     X,d      Modifie la mémoire
ADDA    X,d      Modifie l'accumulateur
```

4.5.2

		Avant		Après	
		(A)	(Z)	(A)	(Z)
LDA	Z,d	123	456	456	456
ADDA	Z,d	123	321	444	321
LDA	Z,d	123	456	456	456
LDA	Z,d	123	456	456	456
ADDA	Z,d	333	123	456	123
STA	Z,d	456	123	456	456

4.5.3

```
LDA     N1,d
ADDA    N2,d
ADDA    N3,d          ;somme
STA     Somme,d
ADDA    Somme,d
ADDA    Somme,d  ;triple somme
```

Chapitre 5

5.3.1 Espacement ou tabulation.

5.3.2

```
X   .WORD  1940
Y   .WORD  0X794
```

5.3.3

```
LDA   X,d
ADDA  4,i
LDX   Y,d
SUBX  12,i
STX   Minus,d
SUBA  Minus,d
ADDA  3,i
```

Chapitre 6

6.7.1 A5 = 0x201AE
déplacement 0x20592 –0x201AE = 0x3E4.

6.7.2 Adresse effective = 0x125A + 0x123 = 0x137D

6.7.3

```
LDX Point,d
LDX 0,x
LDA 0,x
```

 ©2009 Ph. Gabrini

Chapitre 7

7.11.1 Remplacer le `BRNE EnOrdre` par `BRGT EnOrdre`

7.11.2 Remplacer le `CHARO ' ',i` par `CHARO '\n',i`.

7.11.3 Après `BREQ Affiche` ajouter les 5 instructions

```
        CPA  0,i ;
        BRGE Posit ;
        ADDA sommeNeg,d ;
        STA  sommeNeg,d ;
        BR   boucle ;
```

puis ajouter l'étiquette `Posit:` devant `ADDA somme,d ;` et ajouter une déclaration de variable `sommeNeg: .WORD 0 ;` juste avant le `.END`.

7.11.4 Après la première instruction ajouter les instructions suivantes :

```
                LDX  0,i ;
        Suivant: ADDX 1,i ;
                 CPX  10,i ;
                 BRGT Fin ;
```

et enlevez l'ancienne étiquette `Suivant` d'en avant du `DECI`.

7.11.5 Déplacez l'étiquette FinBouc1 sur l'instruction `CHARO '\n',i` et éliminez les trois instructions qui précèdent. Ajoutez l'instruction `LDX 0,i` avant l'étiquette `Boucle2`. Modifiez l'instruction `CPX 0,i` en `CPX TAILLE,i`. Modifiez l'instruction `BRLT FinBouc2` en `BRGE FinBouc2`. Enfin, modifiez l'instruction `SUBX` en `ADDX`.

7.11.6 Après l'instruction d'adresse 0021, ajoutez les instructions suivantes et l'étiquette `Cels` :

```
                CPA     'C',i       ;     if(degrees == 'C'
                BREQ    Cels        ;
                CPA     'f',i       ;         || degres == 'c')
                BREQ    Cels        ;
                STRO    erreur,d    ;
                CHARO   NEWLINE,i   ;
                BR      Fin         ;
        Cels:   LDA     temper,d    ;
```

Ajoutez également la définition de la chaîne de caractères `erreur`.

Chapitre 8

8.6.1.1 Le caractère de fin correspond au troisième cas de chaque état; dans chacun des états il faut donc aller au troisième cas qui est situé après les deux premiers qui occupent chacun 2 octets: on doit donc sauter 4 octets.

8.6.1.2 Chaque état comporte trois cas représentés par trois étiquettes et par conséquent par trois instructions de saut. Chaque instruction de saut occupe 2 octets; il faut donc 6 octets par état.

Chapitre 10

10.2.1 Voir Figure 10.1. On y ajoutera des combinaisons pleines (5 traitements) ou vides de traitements tous différents ou tous semblables.

10.2.3 Sous-programmes pour la vidange partielle.

```
;------- Vidange -------
;  Vidange de mémoire entre deux limites données
;  par l'utilisateur
;  Contenu des mots mémoire donnés en hexadécimal
;  Les limites doivent être données en hexadécimal
;  Appel: empiler adresse message 1
;         empiler adresse message 2
;         empiler adresse titre
;         CALL Vidange
;  registres restaurés
Vcompte: .EQUATE   0         ; compte
Vindice: .EQUATE   2         ; indice
Vbasse:  .EQUATE   4         ; Limite inférieure
Vhaute:  .EQUATE   6         ; Limite supérieure
VVieuxX: .EQUATE   8         ; Sauvegarde registre X
VVieuxA: .EQUATE   10        ; Sauvegarde registre A
VAdRet:  .EQUATE   12        ; Adresse de retour
VTitre:  .EQUATE   14        ; Adresse titre
VAdMes2: .EQUATE   16        ; Adresse message 2
VAdMes1: .EQUATE   18        ; Adresse message 1
                             ;void Vidange(){
Vidange: SUBSP   12,i        ;  espace sauvegarde
         STA     VVieuxA,s   ;
         STX     VVieuxX,s   ;
         CHARO   NEWLINE,i   ;  cout << endl
         STRO    VAdMes1,sf  ;       << "Limite 1 S.V.P ";
         SUBSP   2,i         ;  espace pour nombre lu
         CALL    Hexin       ;  lire la limite
         LDA     0,s         ;
         ADDSP   2,i         ;  désempiler et ranger
         STA     Vbasse,s    ;  la première limite lue
         CHARO   NEWLINE,i   ;  cout << endl
         STRO    VAdMes2,sf  ;       << "Limite 2 S.V.P ";
         SUBSP   2,i         ;  espace pour nombre lu
         CALL    Hexin       ;  lire la limite
         LDA     0,s         ;
         ADDSP   2,i         ;  désempiler et ranger
         STA     Vhaute,s    ;  la deuxième limite lue
         CPA     Vbasse,s    ;  if(haute > basse)
         BRGT    VOK         ;
         LDX     Vbasse,s    ;    Basse <=> Haute
         STA     Vbasse,s    ;
         STX     Vhaute,s    ;
VOK:     CHARO   NEWLINE,i   ;  cout << endl
         STRO    VTitre,sf   ;       << " Adresse    Hexadécimal ";
         LDX     0,i         ;
         STX     Vindice,s   ;  while(true){
VAffiche:CHARO   NEWLINE,i   ;    cout << endl;
         LDA     Vbasse,s    ;
```

 ©2009 Ph. Gabrini

```
          ADDA    Vindice,s   ;    {calcule adresse}
          STA     -2,s        ;
          SUBSP   2,i         ;    empiler valeur
          CALL    Hexout      ;    cout << Adresse,4)
          CHARO   ESPACE,i    ;         << ' '
          CHARO   ESPACE,i    ;         << ' '
          CHARO   ESPACE,i    ;         << ' '
          CHARO   ESPACE,i    ;         << ' '
          LDA     0,i         ;    for(i = 1; I <= 8; I++){
          STA     Vcompte,s   ;
Mot:      LDA     Vbasse,sxf  ;
          STA     -2,s        ;
          SUBSP   2,i         ;      empiler valeur
          CALL    Hexout      ;      cout << mot
          ADDX    2,i        ;       mot suivant
          CHARO   ESPACE,i    ;      cout << ' ';
          LDA     Vcompte,s   ;
          ADDA    1,i        ;
          STA     Vcompte,s   ;
          CPA     8,i         ;
          BRLT     Mot         ;    }// for
          CHARO   ESPACE,i    ;    cout << ' '
          CHARO   ESPACE,i    ;         << ' ';
          LDX     Vindice,s   ;    replacer indice en début de ligne
          LDA     0,i         ;    for(i = 1; I <= 16; I++){
          STA     Vcompte,s   ;
Car:      LDA     0,i         ;
          LDBYTEA Vbasse,sxf  ;      if(car > ' '
          CPA     ' ',i       ;
          BRLT    Anormal     ;
          CPA     '~',i       ;         && car <= '~'
          BRLE    Vnormal     ;
          CPA     'À',i       ;         || car >= 'À')
          BRGE    Vnormal     ;
Anormal:  CHARO   '.',i       ;        cout << car;
          BR      Suivant     ;      else
Vnormal:  CHARO   Vbasse,sxf  ;        cout << '.';
Suivant:  ADDX    1,i         ;      caractère suivant
          LDA     Vcompte,s   ;
          ADDA    1,i         ;
          STA     Vcompte,s   ;
          CPA     16,i        ;
          BRLT    Car         ;    }// for
          STX     Vindice,s   ;
          LDA     Vbasse,s    ;    zone de sortie
          ADDA    Vindice,s   ;
          CPA     Vhaute,s    ;  if(basse + indice > haute) break;
          BRLE    VAffiche    ;  }// while
          LDA     VAdRet,s    ;  adresse retour
          STA     VAdMes1,s   ;  déplacée
          LDA     VVieuxA,s   ;  restaure A
          LDX     VVieuxX,s   ;  restaure X
          ADDSP   18,i        ;  nettoyer pile
          RET0                ;return

;------- Hexin -------
;Lecture d'un nombre hexadécimal dont l'adresse se
```

```
;trouve sur la pile.
hCarac:  .EQUATE   0         ; Caractère lu
hVieuxX: .EQUATE   2         ; Sauvegarde registre X
hVieuxA: .EQUATE   4         ; Sauvegarde registre A
hAdRet:  .EQUATE   6         ; Adresse de retour
hNombre: .EQUATE   8         ; Nombre à lire
                             ;void Hexin(int &N)
Hexin:    SUBSP   6,i        ;  espace local sauvegarde
          STA     hVieuxA,s  ;  sauvegarde A
          STX     hVieuxX,s  ;  sauvegarde X
          LDA     0,i        ;
          STA     hNombre,s  ;  nombre := 0;
Lire:     CHARI   hCarac,s   ;  while(true){
          LDBYTEA hCarac,s   ;    lire caractère
          CPA     'F',I      ;    if(carac > 'F')
          BRGT    FinHex     ;      break;
          CPA     'A',i      ;    else if(carac >= 'A')
          BRGE    Convert    ;      Convertir
          CPA     '9',I      ;    else if(carac > '9')
          BRGT    FinHex     ;      break;
          CPA     '0',I      ;    else if(carac < '0')
          BRLT    FinHex     ;      break;
          SUBA    '0',i      ;    else valeur = carac - '0';
Garde:    LDX     hNombre,s  ;
          ASLX               ;    nombre *= 16;
          ASLX               ;
          ASLX               ;
          ASLX               ;
          STX     hNombre,s  ;    nombre += valeur;
          ADDA    hNombre,s  ;
          STA     hNombre,s  ;
          LDA     0,i        ;    nettoyer pour LDBYTEA
          BR      Lire       ;  }
Convert:  SUBA    'A',I      ;  valeur = carac - 'A'
          ADDA    10,i       ;            + 10;
          BR      Garde      ;
FinHex:   LDA     hVieuxA,s  ;  restaure A
          LDX     hVieuxX,s  ;  restaure X
          RET6               ;}// Hexin;
;Sous-programme Hexout.
;affiche un mot sous la forme de 4 charactères hexadécimaux.
;
hoTemp:   .EQUATE   0        ; Caractère temporaire
hoVieuxX: .EQUATE   2        ; Sauvegarde registre X
hoVieuxA: .EQUATE   4        ; Sauvegarde registre A
hoAdRet:  .EQUATE   6        ; Adresse de retour
hoNombre: .EQUATE   8        ; Nombre à afficher
;
Hexout:   SUBSP   6,i        ;  réserver
          STA     hoVieuxA,s ;  sauvegarde A
          STX     hoVieuxX,s ;  sauvegarde X
          LDA     hoNombre,s ;  A = nombre
          STA     hoTemp,s   ;  sauvegarder mot
          LDBYTEA hoTemp,s   ;  premier caractère
          ASRA               ;  décaler 4 bits
          ASRA               ;
          ASRA               ;
```

 ©2009 Ph. Gabrini

```
          ASRA              ;
          CALL    ViderA    ;  premier caractère hexa
          LDBYTEA hoTemp,s  ;  second 4 bits dans A
          CALL    ViderA    ;  second caractère hexa
          LDA     hoTemp,s  ;  3ème 4 bits dans A
          ASRA              ;  décaler 4 bits
          ASRA              ;
          ASRA              ;
          ASRA              ;
          CALL    ViderA    ;  troisième caractère hexa
          LDA     hoTemp,s  ;  dernier 4 bits dans A
          CALL    ViderA    ;  quatrième caractère hexa
          LDA     hoAdRet,s ;
          STA     hoNombre,s ;
          LDA     hoVieuxA,s ;  restaure A
          LDX     hoVieuxX,s ;  restaure X
          ADDSP   8,i       ; nettoyer pile
          RET0              ;return
;
;Sous-programme de sortie des 4 bits les moins significatifs de A
;
VTemp:    .EQUATE   0       ; Caractère temporaire
VidAdRet: .EQUATE   2       ; Adresse de retour
;
ViderA:   SUBSP   2,i       ;
          ANDA    0xF,i     ;  isoler valeur chiffre hexa
          CPA     9,i       ;  si pas dans 0-9
          BRLE    PrepNb    ;
          SUBA    9,i       ;  convertir nombre en lettre ASCII
          ORA     0x040,i   ;  et préfixe code lettre  ASCII
          BR      Affiche   ;
PrepNb:   ORA     0x030,i   ;  sinon préfixe code nombre ASCII
Affiche:  STBYTEA VTemp,s   ;  For output
          CHARO   VTemp,s   ;
          RET2              ;
```

Chapitre 11

11.2.1.1 Nombres hexadécimaux en représentation réelle sur 32 bits.

a) AB1.234 $\times 16^{3}$ = 1.01010110001001000110100 × 2^{23} = 4B2B1234

b) 0.12ABC34 $\times 16^{15}$ = 1.00101010101111100001101 × 2^{56} = 5B955E1A

c) 64.532A $\times 16^{-8}$ = 1.10010001010011001010100 × 2^{-26} = 32C8A654

d) 0.BCDEF $\times 16^{10}$ = 1.0111100110111101111 × 2^{39} = 533CDEF0

e) A1.B2C3 $\times 16^{-3}$ = 1.01000011011001011000011 × 2^{-5} = 3D21B2C3

11.2.1.2 Nombres hexadécimaux en représentation réelle sur 64 bits.

a) 123456789ABCDEF × 16^{22} = 1.23456789ABCDEF × 16^{36} = 48F23456789ABCDE

b) FEDCBA.9876543210 × 16^{-12} = 3E6FDB97530ECA86

c) 0.123456789AB × 16^{-20} = 3AB23456789AB000

d) 0.00000AECDEF × 16^{8} = 40A5D9BDE0000000

e) 1234567.89ABC $\times 16^{-7}$ = 3FB23456789ABC00

11.2.1.3 Nombres réels en notation scientifique usuelle (base 16).

a) 0A123456 = 2.48D158 × 16^{-27}

b) C643210A = -3.0C8428 × 16^{3}
c) 45AB12C0 = 1.56258 × 16^{3}
d) EFABCDEF = -1.579BDE × 16^{24}
e) FFFFFFFF = -1.FFFFFFE × 16^{32}

11.2.1.4 Nombres réels en notation scientifique usuelle (base 16).

a) B2123456 789ABCDE = -4.8D159E26AF378 × 16^{55}
b) 12345678 9ABCDEF0 = 1.456789ABCDEF0 × 16^{-183}
c) 81234567 81234567 = -0.9A2B3C091A2B38 × 16^{-251}
d) FEDCBA98 76543210 = -1.CBA9876543210 × 16^{251}
e) 6543210F EDCBA987 = 2.6421FDB97530E × 16^{149}

11.2.1.5 Nombres hexadécimaux en nombre décimaux.

a) 0.00A = 0.00244140625
b) 0.1234 = 0.07110595703125
c) 0.A = 0.625
d) 12.AB = 18.66796875
e) AEC.123 = 2796.071044921875

11.2.1.6 Nombres décimaux en nombres hexadécimaux.

a) 0.1250 = 0.2
b) 0.22900390625 = 0.3AA
c) 0.9375 = 0.F
d) 9876.5 = 2694.8
e) 8192.0940185546875 = 2000.F0B

11.2.1.7: X+Y et X*Y

C7A40000 + 402C0000 = CFA40158
C7A40000 * 402C0000 = C85C6000

Chapitre 12

12.5.7.1 La boucle de décalage `loop` place le bit d'adressage de l'instruction interrompue en position pour le masque ; si le mode d'adressage est 101, le masque produit sera 00100000. On doit utiliser l'appel `CALL prntMsg`, car l'instruction `STRO` est elle même sujette à engendrer une interruption, ce qu'on doit éviter puisque le système de traitement des interruptions de PEP 8 ne permet pas d'interrompre une interruption…

12.5.7.2 Après l'étiquette `addrN`, il faut répéter l'instruction `LDX 0,x` puisque l'adressage est indirect et la première instruction ne ramène que l'adresse de l'opérande. Peu après l'étiquette `addrS` la valeur placée dans X par l'instruction `LDX 0,x` est le déplacement relatif au pointeur de pile. Peu après l'étiquette `addrSX` le contenu de X après l'instruction `ADDX` est la somme du déplacement sur la pile et du contenu du registre X au moment de l'interruption, et après la seconde instruction la valeur de X est augmentée de la valeur du pointeur de pile avant l'interruption. Après l'étiquette `addrSXF` après exécution de la seconde instruction `LDX 0,x`, X contient l'adresse d'un vecteur qui était rangée sur la pile.

12.5.7.3 Les symboles `init`, `sign` et `digit` sont des constantes utilisées comme valeur de l'état (state). L'instruction `ANDA 0x000F,i` permet de passer du code ASCII du caractère numérique à la valeur numérique correspondante. Après exécution de l'instruction `STX total,s`, total comprend la valeur numérique du chiffre lu. On va à `deciErr` si on ne trouve pas d'espace ou de saut de ligne parce qu'on cherche encore le premier caractère de la valeur et on a épuisé toutes les possibilités. La comparaison `CPA 0x8000,i` sert à traiter le cas particulier de la valeur –32768 qui ne déclenche pas un débordement.

 ©2009 Ph. Gabrini

Peu après l'étiquette `setNZ`, on effectue un ET logique avec la valeur 1 parce qu'on veut laisser la valeur du code de condition C comme elle était. À l'étiquette `storeFl` on range la valeur sur la pile système à l'endroit où les codes de condition ont été sauvegardés ; la valeur des codes de condition est mise à jour.

12.5.7.4 À l'étiquette `printDgt`, l'instruction `ORX 0x0030,i` permet de convertir une valeur numérique <10 en son équivalent caractère ASCII (tous les codes ASCII des chiffres décimaux commencent par 3).

12.5.7.5 L'instruction `ADDSP` située à l'adresse `FFDE` est là pour nettoyer la pile du paramètre empilé pour l'appel à `prntMsg`.

Chapitre 13

13.1.4 1 Théorème 6 : X+X'Y = X+Y

Preuve	X+X'Y	= (X+XY)+X'Y	théorème 1
		= X+XY+X'Y	
		= X+Y(X+X')	X'+X=1
		= X+Y(1)	
		= X+Y	

13.1.4 2 Théorème 7 : (X+Y)(X'+Z) = XZ+X'Y

Preuve	(X+Y)(X'+Z)	= X'X+XZ+X'Y+YZ	multiplication
		= XZ+X'Y+YZ	XX'=0
		= XZ+X'Y	théorème 5

13.1.4 3 Théorème 8 : (X+Y)(X'+Z)(Y+Z) = (X+Y)(X'+Z)

Preuve	(X+Y)(X'+Z)(Y+Z)	= (XZ+X'Y)(Y+Z)	théorème 7
		= XYZ+XZZ+X'YY+X'YZ	
		= XYZ+XZ+X'Y+X'YZ	YY=Y
		= XZ(Y+1)+X'Y(1+Z)	
		= XZ+X'Y	
		= (X+Y)(X'+Z)	théorème 7

13.1.4 4 Théorème 9 : (XYZ)' = X'+Y'+Z'

Preuve : Supposons que le théorème soit vrai et testons ses conséquences. Si X'+Y'+Z' est le complément de XYZ, avec les axiomes de base de l'algèbre de Boole on a : (X'+Y'+Z')(XYZ) = 0 et (X'+Y'+Z')+(XYZ) = 1. Reprenons chacune de ces affirmations et prouvons les.

(X'+Y'+Z')(XYZ)	= X'XYZ+Y'XYZ+Z'XYZ	
	= X'X(YZ)+Y'Y(XZ)+Z'Z(XY)	
	= 0	
(X'+Y'+Z')+(XYZ)	= YZ(X)+X'+Y'+Z'	réarranger les termes
	= YZ+X'+Y'+Z'	AB+B'=A+B'
	= (Y'+YZ)+X'+Z'	réarranger les termes
	= Y'+Z+Z'+X'	
	=Y'+1+X' = 1	Z+Z'=1

13.1.4 5 Simplifier X'YZ'+X'YZ+XY'Z+XYZ

X'YZ'+X'YZ+XY'Z+XYZ = X'Y(Z'+Z)+XZ(Y'+Y)

= X'Y(1)+XZ(1) car Z'+Z=1

13.1.4 6 Simplifier ((X'Y)'(XZ)')'

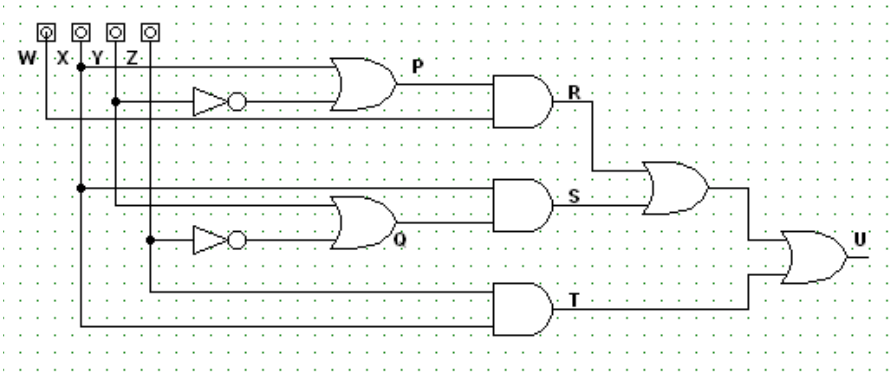
((X'Y)'(XZ)')'	= ((X'Y)')'+((XZ)')'	Théorème 8
	= X'Y+XZ	car (F')'=F

13.1.4 7 Simplifier (W+X+YZ)(W'+X)(X'+Y)

(W+X+YZ)(W'+X)(X'+Y)	= (WW'+W'X+W'YZ+WX+XX+XYZ)(X'+Y)
	= (W'X+W'YZ+WX+X+XYZ)(X'+Y)
	= (X+W'YZ)(X'+Y)
	= XX'+XY+W'X'YZ+W'YYZ
	= XY+W'X'YZ+W'YZ
	= XY+W'YZ(X'+ 1)
	= XY+W'YZ

13.3.2 1

Étant donné le diagramme ci-dessous, établissez une table de vérité pour les valeurs des quatre variables et les valeurs de P, Q, R, S, T, U et V. Donnez les expressions logiques correspondantes.

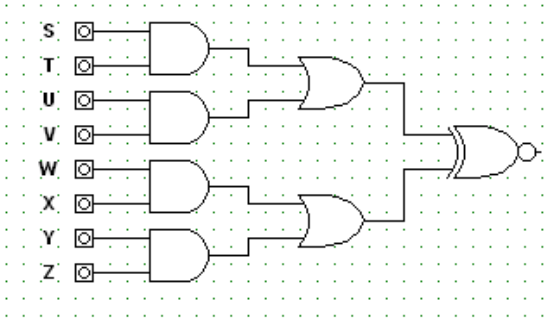


W	X	Y	Z	P	Q	R	S	T	U
0	0	0	0	1	1	0	0	0	0
0	0	0	1	1	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0
0	0	1	1	0	1	0	0	0	0
0	1	0	0	1	1	0	1	0	1
0	1	0	1	1	0	0	0	1	1
0	1	1	0	1	1	0	1	0	1
0	1	1	1	1	1	0	1	1	1
1	0	0	0	1	1	1	0	0	1
1	0	0	1	1	0	1	0	0	1
1	0	1	0	0	1	0	0	0	0
1	0	1	1	0	1	0	0	0	0
1	1	0	0	1	1	1	1	0	1
1	1	0	1	1	0	1	1	1	1
1	1	1	0	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1

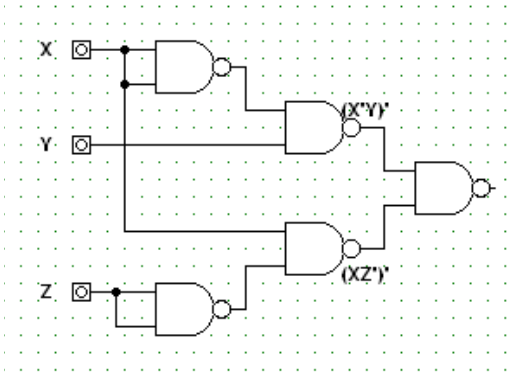
P = X+Y'; Q = Y+Z' ; R = WP ; S = QX ; T = XZ ; U = R+S+T.

 ©2009 Ph. Gabrini

13.3.2 2 F= (ST+UV+WX+YZ)'.



13.3.2 3 F= X'Y+XZ' = (F')' = ((X'Y+XZ')')' = ((X'Y)'(XZ')')'



13.3.2 4

W0 = A1'A0B1'B0+A1'A0B1B0+A1A0B1'B0+A1A0B1B0
= A1'A0B0(A1'+A1)+A1A0B0(B1'+B1)
= A1'A0B0+A1A0B0
= A0B0(A1'+A1)
= A0B0

W1 = A1'A0B1B0'+A1'A0B1B0+A1A0'B1'B0+A1A0'B1B0+A1A0B1'B0+A1A0B1B0'
=A1'A0B1(B0'+B0)+A1A0'B0(B1'+B1)+A1A0B1'B0+A1A0B1B0'
= A1'A0B1+A1A0'B0+A1A0B1'B0+A1A0B1B0'
= A0B1(A1'+A1B0')+A1B0(A0'+A0B1')
= A0B1(A1'+B0')+A1B0(A0'+B1')
= A1'A0B1+A0B1B0'+A1A0'B0+A1B1'B0

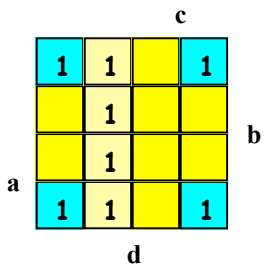
W2 = A1A0'B1B0'+A1A0'B1B0+A1A0B1B0'
= A1A0'B1(B0'+B0)+A1A0B1B0'
= A1A0'B1+ A1A0B1B0'
= A1B1(A0'+A0B0')
= A1B1(A0'+B0')
= A1A0'B1+A1B1B0'

W3 = A1A0B1B0

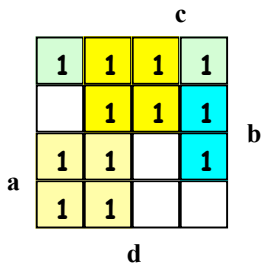
13.3.3.1.1

Montrez la minimisation du circuit a'b'c'd'+a'b'c'd+a'b'cd'+a'bc'd+ab'c'd'+ab'c'd+ab'cd'+abc'd au moyen de la table de Karnaugh.

c'd+b'd'



13.3.3.1.2 Montrez la minimisation du circuit suivant par table de Karnaugh a'b'c'd'+a'b'c'd+a'b'cd'+a'b'cd+a'bc'd+a'bcd'+a'bcd+ab'c'd'+ab'c'd+abc'd'+abc'd+abcd'.

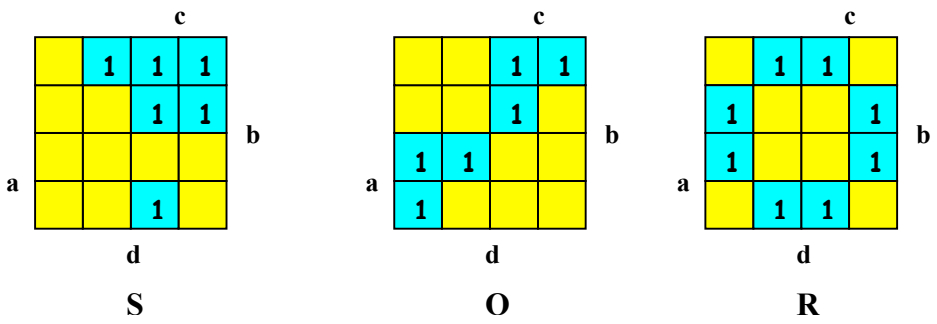


ac'+a'd+a'b'+bcd' Attention ! il ne faut pas prendre un autre découpage qui semble logique, mais prend plus de termes : ac'+a'c+c'd+a'b'+bcd'. Dans certains cas il faut étudier la situation plus en détail.

13.3.3.1.3 Un circuit logique possède quatre entrées A, B, C et D; ces données représentent deux paires de bits (A,B) et (C,D). On soustrait les bits (C,D) des bits (A,B) pour donner un résultat (Q,R) et un signe S pour le résultat (1 si négatif). Établissez la table de vérité pour ce circuit et simplifiez les expressions des trois sorties par tables de Karnaugh.

A	B	C	D	Val	S	Q	R
0	0	0	0	0	0	0	0
0	0	0	1	-1	1	0	1
0	0	1	0	-2	1	1	0
0	0	1	1	-3	1	1	1
0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	0
0	1	1	0	-1	1	0	1
0	1	1	1	-2	1	1	0
1	0	0	0	2	0	1	0
1	0	0	1	1	0	0	1
1	0	1	0	0	0	0	0
1	0	1	1	-1	1	0	1
1	1	0	0	3	0	1	1
1	1	0	1	2	0	1	0
1	1	1	0	1	0	0	1
1	1	1	1	0	0	0	0

 ©2009 Ph. Gabrini



Donc S = a'c+a'b'd+b'cd; Q = a'b'c+a'cd+abc'+ac'd'; R = b'd+bd'.

13.3.5.1 abc+ab'c+a'bc+a'b'c+a'b'c'

Termes et chaînes	Étape 3	Étape 4
1 a'b'c' 000	(1,2) a'b' 00-	(2,3,4,5) c --1
2 a'b'c 001	(2,3) a'c 0-1	
3 a'bc 011	(2,4) b'c -01	
4 ab'c 101	(3,5) bc -11	
5 abc 111	(4,5) ac 1-1	

À l'étape 3, nous avons combiné tous les produits de la première colonne de la table. À l'étape 4, nous n'avons pu combiner que les 4 derniers termes de la colonne (en jaune pâle); il reste donc un terme de cette colonne non combiné. Le terme de la troisième colonne est seul et ne peut être combiné. Par conséquent, nous terminons l'étape 5 avec deux termes non utilisés, soit a'b' et c. La table ci-dessous montre quels termes sont couverts par les termes réduits retenus.

	abc	ab'c	a'bc	a'b'c	a'b'c'
c	X	X	X	X	
a'b'				X	X

Réponse: c+a'b'

13.3.5.2 abcd+abc'd+abc'd'+ab'cd+a'bcd+a'bc'd+a'bcd'+a'b'c'd

Termes et chaînes	Étape 3	Étape 4
1 a'b'c'd 0001	(1,3)a'c'd 0-01	(3,6,5,8)bd -1-1
2 a'bcd' 0110	(2,5) a'bc 011-	
3 a'bc'd 0101	(3,5) a'bd 01-1	
4 abc'd' 1100	(3,6) bc'd -101	
5 a'bcd 0111	(4,6) abc' 110-	
6 abc'd 1101	(5,8) bcd -111	
7 ab'cd 1011	(6,8) abd 11-1	
8 abcd 1111	(7,8) acd 1-11	

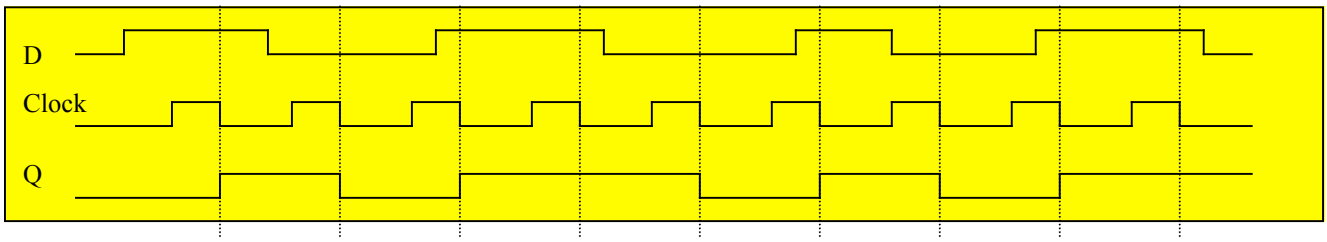
À l'étape 3, nous avons combiné tous les produits de la première colonne de la table. À l'étape 4, nous n'avons pu combiner que 4 termes de la colonne (en jaune pâle); il reste donc quatre termes de cette colonne non combinés. Le terme de la troisième colonne est seul et ne peut être combiné. Par

conséquent, nous terminons l'étape 5 avec cinq termes non utilisés, soit bd, abc', a'bc, acd et a'c'd. La table ci-dessous montre quels termes sont couverts par les termes réduits retenus.

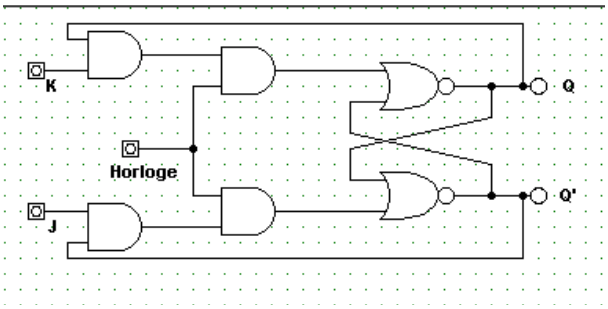
	abcd	abc'd	abc'd'	ab'cd	a'bcd	a'bc'd	a'bcd'	a'b'c'd
bd	X	X			X	X		
abc'		X	X					
a'bc					X		X	
acd	X			X				
a'c'd						X		X

L'expression finale est donc: abc'+a'bc+acd+a'c'd

13.4.4 1 Diagramme temporel pour la bascule D.



13.4.4 2 Circuit d'une bascule J-K construite à partir d'une bascule S-R.



 ©2009 Ph. Gabrini