Chapitre 6

Les bus

Nous avons vu comment réaliser des opérations arithmétiques primitives et comment réaliser les registres (de travail), autrement dit la mémoire. Nous allons voir dans ce chapitre, d'une part, comment réaliser les transferts entre registres et, d'autre part, comment réaliser les opérations arithmétiques primitives avec un seul circuit par opération, par exemple l'addition de n'importe quelle paire de registres avec un seul additionneur.

Nous allons voir également comment ceci nous permet de concevoir une calculatrice (électronique) ayant des entrées-sorties rudimentaires.

6.1 Réalisation des transferts

6.1.1 Transfert direct entre registres

On peut transférer le contenu d'un registre B à un registre A, sans détruire le contenu de B, soit par deux impulsions successives comme montré à la figure 6.1, l'une de mise à zéro et l'autre de transfert des 1.

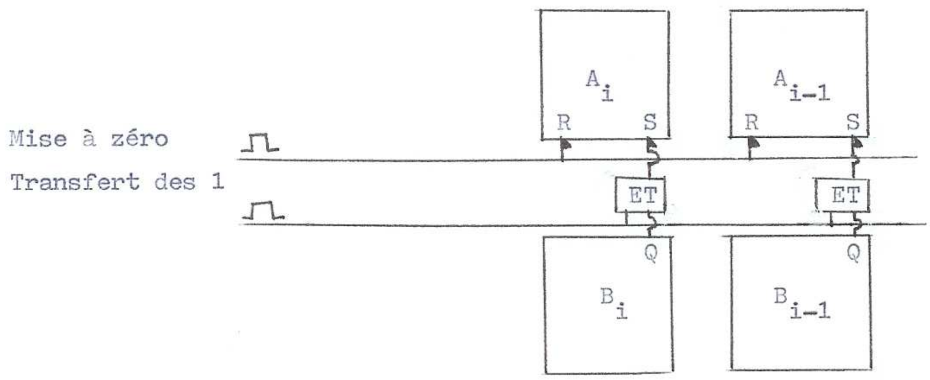

FIGURE 6.1 – Transfert entre registres par deux impulsions

On peut aussi transférer les 0 et les 1 simultanément (figures 6.2 et 6.3).

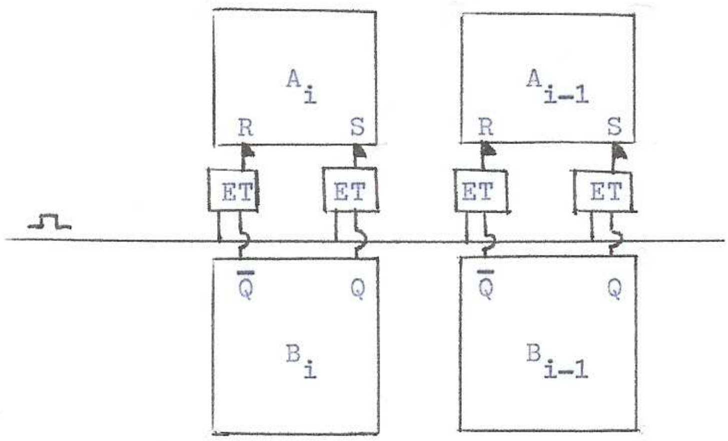

FIGURE 6.2 – Premier système de transfert entre registres à entrée forcée

Dans les trois cas on a besoin d'une impulsion (et même de deux dans le premier cas) pour effectuer le transfert, disons pour indiquer à quel moment effectuer le transfert. On l'appelle **impulsion de transfert**.

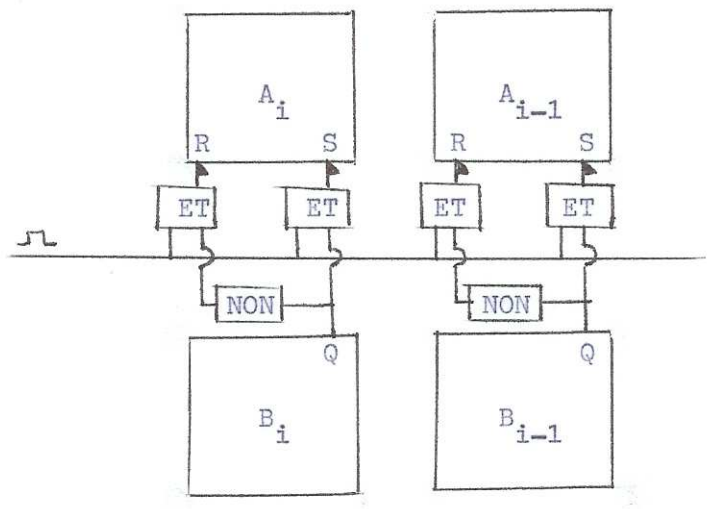

FIGURE 6.3 – Deuxième système de transfert entre registres à entrée forcée

Le principe de ce transfert de registre à bistables vers registre à bistables peut également s'effectuer de registre à interrupteurs vers registre à bistable et de registre à bistables vers registre à ampoules.

6.1.2 Le problème topologique de la liaison entre registres

La première idée pour réaliser les transferts est de tout simplement relier un registre à tous les autres registres. Mais ceci exige un câblage considérable. En effet s'il y a m registres de largeur n, pour chaque registre il faut $(m-1) \times n$ fils pour envoyer de l'information vers les $m-1$ autres registres plus $m-1$ fils pour les impulsions de transfert correspondantes. Ceci fait $m \times n$ fils. Il en faut autant pour chacun des m registres, ce qui fait $m^2 \times n$ fils. Pour cent registres de largeur huit, cela fait 80 000 liaisons à réaliser.

Pour simplifier le câblage à réaliser, on s'inspire des centraux téléphoniques.

Le téléphone électrique est déployé de façon industrielle très peu de temps après sa conception, dans la seconde moitié du XIX^e siècle : il suffit de deux micros, de deux écouteurs, d'une batterie et de deux fils pour une communication d'un point à un autre (nous avons vu, en fait, qu'il en suffit d'un seul, en utilisant la terre pour le deuxième). Dès les essais terminés, des installations fixes sont réalisées entre certains endroits. Le téléphone connaît un engouement immédiat : beaucoup de gens veulent communiquer par ce moyen. Si A veut pouvoir communiquer avec B, C et D alors il faut trois câbles partant de chez lui. Certaines villes furent envahies par les câbles téléphoniques (aériens).

L'inventeur du téléphone, Graham BELL, a alors l'idée des *centraux téléphoniques*. Chaque personne se relie, par un câble, à un seul endroit, appelé *central téléphonique* CT. Lorsqu'une personne A veut parler à une personne B, elle appelle le seul endroit qu'elle peut appeler, c'est-à-dire le central, et indique à la personne au bout du fil (appelée *opératrice*, car les garçons employés à l'origine sont vite remplacés par des femmes) qu'elle veut parler à B. L'opératrice la relie alors, grâce à un tableau, à B (si B n'est pas déjà en train de téléphoner) et la conversation peut commencer. Ainsi sont nées les compagnies de téléphone.

La configuration des liaisons entre les registres est inspirée de l'exemple téléphonique, où les centraux sont remplacés par des *bus*. Dans notre exemple, on passe de 80 000 liaisons à 800, ce qui est un progrès considérable.

6.1.3 Les bus

6.1.3.1 Notion de bus

Un **bus**, *ou* **ligne omnibus**, *est un ensemble de fils sur chacun desquels une information binaire peut transiter sous forme de tension.*
La **largeur** *d'un bus est le nombre de fils (et donc de bits maintenus).*

Nous représenterons toujours un bus schématiquement par un seul fil et non par plusieurs fils, ce qui est suffisant pour comprendre ce qui se passe.

6.1.3.2 Bus des données

Même si cela n'est pas clair pour l'instant, nous verrons qu'un ordinateur comporte en général trois bus : le bus des données, le bus des adresses et le bus des instructions.

En tous les cas ce qui est clair c'est qu'un (premier) bus est relié aux registres d'entrée, aux registres de sortie et aux registres de travail. On l'appelle le **bus des données** pour le distinguer d'autres bus éventuels.

6.1.3.3 Liaisons registres d'entrée – bus des données

Tous les registres d'entrée sont reliés au bus des données suivant le shéma de la figure 6.4, à l'aide d'un circuit combinatoire de transfert d'entrée `TE`.

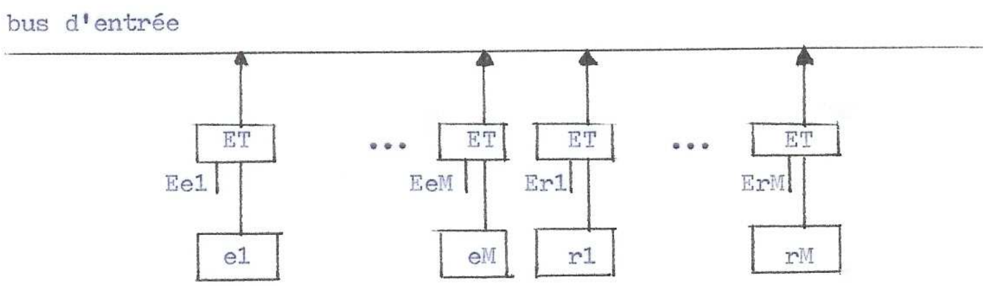

FIGURE 6.4 – Liaisons registres d'entrée – bus des données

Si on prend le cas d'une largeur des données de n, ce circuit de transfert `TE` possède $n+1$ entrées : les n sorties Q_1 à Q_n du registre ainsi qu'une entrée pour le signal d'impulsion. Il a n sorties qui correspondront aux n bits du bus.

Le circuit de transfert est tout simplement constitué de n portes ET : les deux entrées de la i-ième porte sont reliées au canal d'impulsion et à Q_i ; la sortie est reliée au i-ième fil du bus. Le canal d'impulsion est donc utilisé n fois.

Notons `E3` (pour *Entrée*) le canal d'impulsion pour le registre e_3. À chaque instant un seul canal d'impulsion au plus peut se trouver à un niveau haut. Lorsque le canal d'impulsion `E3`, par exemple, est traversé par une impulsion, appelée **signal d'échantillonnage**, en temps voulu alors le bus contient désormais, et pour un temps suffisamment long pour qu'on puisse faire quelque chose de cette information, le contenu du registre d'entrée e_3.

6.1.3.4 Liaisons registres de sortie – bus des données

De même tous les registres de sortie sont reliés au bus des données suivant le shéma de la figure 6.5.

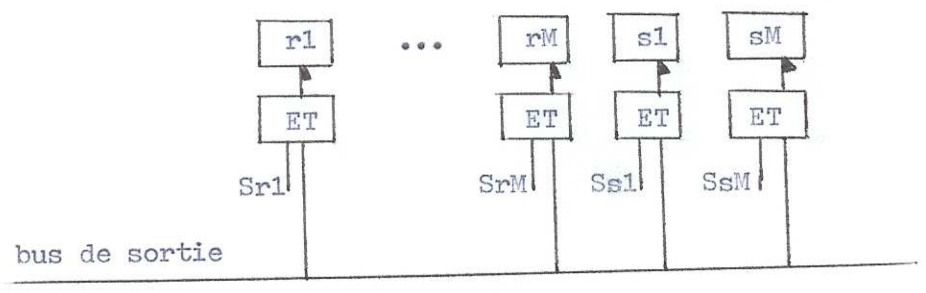

FIGURE 6.5 – Liaisons registres de sortie – bus des données

Le circuit combinatoire de transfert de sortie TS possède $n + 1$ entrées reliées aux n fils du bus ainsi qu'une entrée pour le signal d'impulsion. Il a $2 \times n$ sorties reliées aux entrées R_i et S_i des bistables RS constituant le registre.

Cette fois le circuit de transfert est tout simplement constitué de $2 \times n$ portes ET et de n inverseurs si on prend le cas du deuxième système de transfert à entrée forcée. Le canal d'impulsion est donc utilisé $2 \times n$ fois.

Nous noterons **Si** le canal d'impulsion correspondant au registre r_i.

6.1.3.5 Liaisons bus des données – registres de travail

Les registres de travail sont reliés au bus des données de deux façons, comme le schématise la figure 6.6, à la fois comme s'il s'agissait d'un registre d'entrée et d'un registre de sortie.

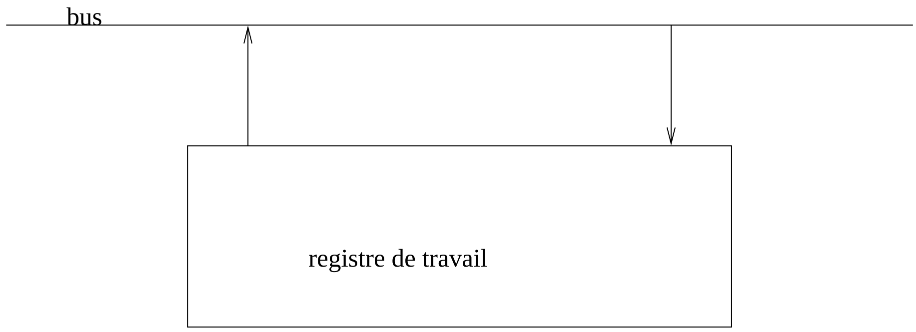

FIGURE 6.6 – Liaisons bus–registres de travail

Notons ERi (pour *Entrée du Registre de travail*) le canal d'impulsion permettant de transmettre l'information du bus des données au registre r_i et SRi (pour *Sortie du Registre de travail*) le canal d'impulsion permettant de transmettre l'information du registre r_i au bus des données.

6.1.4 Réalisation des transferts *via* un bus

Supposons que l'on veuille transférer l'information de e_3 dans r_5, par exemple. Ceci s'effectue en deux étapes :

- On émet une impulsion sur le canal `E3`, en abrégé une impulsion `E3`. Ceci a pour effet de transmettre l'information du registre e_3 au bus des données.
- On émet ensuite une impulsion `SR5`. Ceci a pour effet de transmettre l'information du bus des données vers r_5 et donc en définitive de l'information de e_3 vers r_5.

6.1.5 Pupitre de commande

Le principe du transfert est donc simple. Reste à savoir comment l'utilisateur peut envoyer les impulsions.

On peut, dans un premier temps, imaginer un **pupitre de commande** pilotant m registres d'entrée, m registres de travail et m registres de sortie, tous de largeur n. Celui-ci doit comprendre $m \times n$ interrupteurs pour les registres d'entrée, $m \times n$ lampes pour les registres de sortie et $4 \times m$ boutons poussoirs pour les impulsions. Ne riez pas ! n'est-ce pas ce que vous faites lorsque vous utilisez votre calculettre pour des calculs complexes ?

Dans ce cas un opérateur humain est chargé d'exécuter le programme pas à pas. L'exécution d'un programme est assez long mais moins long que si on l'exécute complètement à la main car une addition s'effectuera en une fraction de seconde.

6.2 Réalisation des instructions primitives

Nous venons un peu d'anticiper dans la mise en œuvre avec notre pupitre de commande puisque nous avons seulement indiqué comment effectuer les transferts mais pas les opérations arithmétiques primitives. Voyons donc maintenant ce qu'il en est pour elles.

6.2.1 L'unité arithmétique

Pour mettre en œuvre les opérations arithmétiques élémentaires que l'on a choisi (et les opérations logiques telles que le ET et le OU bit à bit), le bus des données est relié à une **unité arithmétique** (**ALU** pour l'anglais *Arithmetical and Logical Unit*) comme le montre la figure 6.7.

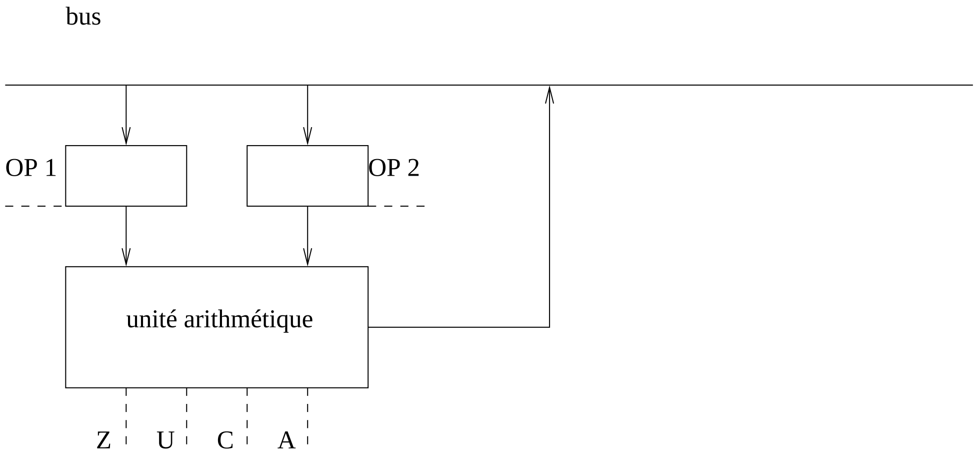

FIGURE 6.7 – L'unité arithmétique

Cette unité arithmétique est constituée de deux registres nouveaux, appelés **opérande un** et **opérande deux**, des circuits combinatoires permettant de réaliser les opérations élémentaires choisies et de six canaux d'impulsions appelés `OP1`, `OP2`, `Z`, `U`, `C` et `A` :

- L'impulsion `OP1` (pour *OPérande 1*) permet de transférer le contenu du bus des données dans le registre `opérande un`.
- L'impulsion `OP2` (pour *OPérande 2*) permet de transférer le contenu du bus des données dans le registre `opérande deux`.
- L'impulsion `Z` (pour *Zéro*) permet de transférer 0 sur le bus des données, ce qui sert à réinitialiser les registres.
- L'impulsion `U` (pour *Un*) permet de transférer 1 sur le bus des données.
- L'impulsion `C` (pour *Complémentation*) permet de transférer l'inverse (bit à bit) du contenu du registre `OP1` sur le bus des données.
- L'impulsion `A` (pour *Addition*) permet de transférer dans le bus des données la somme des contenus des registres `opérande un` et `opérande deux`.

6.2.2 Réalisation des opérations

Considérons maintenant un calculateur avec un pupitre de commande possédant m registres d'entrée, m registres de travail et m registres de sortie, tous de largeur n, puis $m \times n$ interrupteurs pour les registres d'entrée, ainsi que $m \times n$ lampes pour les registres de sortie et $4 \times m$ boutons poussoirs pour les impulsions de transfert mais aussi six nouveaux boutons poussoirs pour déclencher les impulsions `OP1`, `OP2`, `Z`, `U`, `C` et `A`.

Supposons que l'on veuille additionner trois entiers. Ceci s'effectue en quatorze étapes :

- On place le premier entier sur le registre d'entrée e_1 en basculant jusqu'à n interrupteurs.
- On place le deuxième entier sur le registre d'entrée e_2 en basculant jusqu'à n interrupteurs.
- On place le troisième entier sur le registre d'entrée e_3 en basculant jusqu'à n interrupteurs.
- On appuie sur le bouton `E1`, ce qui a pour effet de transmettre l'information du registre e_1 sur le bus des données.
- On appuie sur le bouton `OP1`, ce qui a pour effet de transmettre cette information dans le registre op_1.
- On appuie sur le bouton `E2`, ce qui a pour effet de transmettre l'information du registre e_2 sur le bus des données.
- On appuie sur le bouton `OP2`, ce qui a pour effet de transmettre cette information au registre op_2.
- On appuie sur le bouton `A`, ce qui a pour effet d'additionner les deux premiers entiers et d'en placer le résultat sur le bus des données.
- On appuie sur le bouton `OP1`, ce qui a pour effet de transmettre cette somme partielle dans le registre op_1.
- On appuie sur le bouton `E3`, ce qui a pour effet de transmettre l'information du registre e_3 dans le bus des données.
- On appuie sur le bouton `OP2`, ce qui a pour effet de transmettre cette information dans le registre op_2.
- On appuie sur le bouton `A`, ce qui a pour effet d'additionner les trois entiers et d'en placer le résultat sur le bus des données.
- On appuie sur le bouton `S1`, ce qui a pour effet de transmettre le résultat dans le registre s_1.
- On lit le résultat (en binaire) sur le registre de sortie un.

On peut se demander quel est l'intérêt d'un tel calculateur de nos jours. De tels calculateurs électroniques ont effectivement existé. Leur intérêt ne résidait évidemment pas dans les étapes de transfert. L'intérêt se comprend lorsqu'on appuie deux fois sur le bouton d'addition. Pour une largeur de bus assez grande, effectuer une addition à la main prendrait beaucoup de temps alors que dans un calculateur électronique elle est effectuée rapidement et sans risque d'erreur.

De nos jours il s'agit d'une étape dans la conception d'un calculateur électronique ou, mieux, dans celle d'un ordinateur.

6.3 Historique

6.3.1 Calculateurs à pupitre de commande

On peut utiliser un calculateur électronique, l'analogue de nos calculettes quatre opérations actuelles, implémentant uniquement quelques opérations mais effectuant celles-ci à une vitesse bien supérieure à celle des calculateurs mécaniques de l'époque. Les premiers calculateurs électro-mécaniques et électroniques sont réalisées avec un pupitre de commande.

Nous avons vu qu'ATANASOFF a conçu le premier calculateur électronique, aux États-Unis, en 1939. Au printemps 1942, ATANASOFF et BERRY achèvent la réalisation d'un tel calculateur, connu sous le nom de ABC (*Atanasoff-Berry Computer*). L'opérateur le contrôle, étape par étape, à partir d'une console. Atanasoff

Les opérateurs de l'ENIAC (1945) disposent de trois « tables de fonctions », constituées par les panneaux mobiles, pour introduire soit des nombres (ce qui nous intéressera ici), soit des instructions dans le calculateur. Chaque table de fonction peut enregistrer 104 informations à 14 positions (un nombre de 12 chiffres et son signe, ou deux nombres de six chiffres et deux signes). ENIAC

Il faut aller devant le tableau de commande et introduire, chiffre après chiffre, les nombres (ou les instructions) en tournant à la main des commutateurs à cadrans : 4 368 commutateurs à positionner pour la mise en œuvre complète des trois tables ! Mais après ces fastidieuses opérations initiales, les circuits calculateur de l'ENIAC peuvent accéder à un nombre (ou une instruction) en un millième de seconde. L'efficacité de l'ENIAC repose donc largement sur la vitesse de transfert des données et des instructions entre ses différentes unités.

La sortie des résultats se fait par lecture directe des nombres affichés par de petites lampes témoins sur le panneau frontal des accumulateurs. Cette lecture n'a de sens que lorsqu'un accumulateur a fini de travailler.

Nous venons de décrire une façon d'entrer et de lire les données. En fait les utilisateurs peuvent également fournir des données à l'ENIAC par l'intermédiaire de cartes perforées lues dans un lecteur IBM à la cadence de 120 cartes à la minute et obtenir les résultats sur des cartes perforées grâce à une perforatrice IBM, à la vitesse de 100 cartes à la minute.

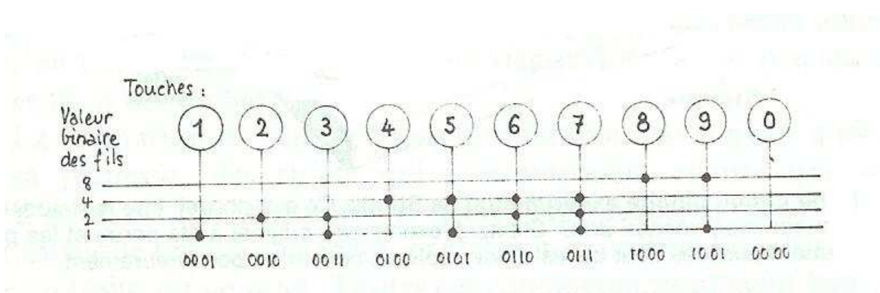

FIGURE 6.8 – Transcodage décimal–binaire ([Lig-87], p. 228)

6.3.2 Transcodage décimal–binaire

Pour simplifier la conception du calculateur que nous avons décrit, nous avons supposé que l'utilisateur entre les entiers en binaire bit à bit et lit les résultats également en binaire. Incontestablement le binaire constitue un obstacle majeur pour des utilisateurs ordinaires. Comment tourner cette difficulté ?

STIBITZ s'est vite rendu compte qu'une machine à calculer décimale est bien trop complexe à réaliser. Durant tout l'hiver 1937 et le printemps 1938, il cherche une solution simple. Puis, soudain, il a l'idée de se rabattre sur un codage éprouvé, utilisé par les ingénieurs en téléphonie depuis 1930 environ : le « décimal codé binaire », en abrégé DCB. Le DCB permet de conserver les avantages découlant de la simplicité inhérente au binaire, sans en supporter les deux inconvénients principaux : la longueur de codes composés de 0 et de 1 et l'obligation de convertir constamment les nombres de décimal en binaire, puis de binaire en décimal à la main.

Un simple raccordement préétabli entre les touches décimales d'un clavier et un ou plusieurs des fils binaires, comme montré sur la figure 6.8, régle tout le problème : l'opérateur travaille en décimal, la machine en mode binaire.

6.4 Bibliographie

[Lig-87] LIGONNIÈRE, Robert, **Préhistoire et histoire des ordinateurs**, Robert Laffont, 1987, 356 p.

6.5 Exercices

Exercice 1.- *Imaginons un calculateur avec* 2 000 *registres de largeur* 8 (*soit* 2 KiO).

- 1^o) *Combien faut-il réaliser de liaisons si on n'utilise pas de bus ?*

- 2^o) *Combien faut-il réaliser de liaisons si on utilise un bus des données ?*

Exercice 2.- Dans le calculateur que nous avons décrit, l'unité arithmétique comprend deux registres car les opérations arithmétiques élémentaires sont unaires et binaires.

Combien faut-il prévoir de registres a priori *si on veut l'opération (ternaire) comme opération élémentaire ?*

Exercice 3.- Dans les exemples de programmes que nous avons donnés, on n'utilise pas d'autres registres de travail que les registres de l'unité arithmétique.

Décrivez le programme (mis en œuvre par un opérateur sur le tableau de commande) pour réaliser ? ? ? ? ?

Votre programme utilise-t-il des registres de travail ?