Chapitre 7

Enregistrement des programmes séquentiels

Nous avons vu comment mettre en œuvre à la main un programme sur un calculateur électromécanique ou électronique grâce à un pupitre de commande. Cette interface devrait vous être familière car c'est encore celle des calculatrices, même scientifiques. Les entrées-sorties sont améliorées certes car on utilise des touches pour les chiffres décimaux et non pour les chiffres binaires et une ligne d'écran (*display* en anglais) permet de voir le résultat, en décimal également. Les calculettes scientifiques ont un intérêt pratique puisqu'il s'en vend un grand nombre par an

Cette façon de faire présente cependant une difficulté. Il faut bien reconnaître que taper cent fois la même suite d'opérations (pour calculer la valeur d'une fonction en une centaine de points, par exemple), en appuyant sur les touches adéquates dans un ordre donné, est lassant et peut conduire à des erreurs dues à des inattentions.

Une première amélioration est apportée à notre modèle précédent de calculateur en prévoyant la mise en place d'un programme « en dur », évitant d'indiquer à la main à chaque fois la suite d'instructions à effectuer.

Commençons par décrire l'enregistrement d'un premier type de programmes : les suites d'instructions élémentaires. On parlera dans le chapitre suivant des ruptures de séquence.

7.1 Codage et décodage des instructions primitives

7.1.1 Codage des instructions primitives

Pour enregistrer les instructions en mémoire, comme pour toute information, il faut **coder** les instructions. On parle de **langage machine** pour la façon de coder les instructions en binaire. Pour une meilleure compréhension par un être humain, on commence presque toujours par écrire écrire les instructions dans un langage intermédiaire, appelé alors **langage symbolique**.

7.1.1.1 Langage symbolique

En langage symbolique, une instruction primitive est constituée de deux parties : la nature de l'instruction, suivie éventuellement d'une **adresse**, c'est-à-dire du numéro du registre concerné. Choisissons, pour notre langage symbolique, de représenter la nature de l'instructions par trois lettres majuscules, représentant une dénomination mnémotechnique de ce qu'elle doit effectuer. Le *mnémonyme* est éventuellement suivi d'un nombre, l'adresse. Par exemple :

```
ENT 3
```

(pour *ENTrée*) signifiera :

`A :=` e_3

où `A` est un registre spécial, appelé **accumulateur**.

Les dénominations mnémotechniques que nous utiliserons sont les suivantes :

Représentation	Signification	Explication du mnémonyme
ENT n	A $:= e_n$	ENTrée
SOR n	$s_n :=$ A	SORtie
CHA n	A $:= r_n$	CHArger
CHC n	A $:= \overline{r_n}$	CHarger Complémentaire
RNG n	$r_n :=$ A	RaNGer
ZER	A $:= 0$	ZÉRo
UNI	A $:= 1$	UNIté
ADD n	A $:=$ A $+ r_n$	ADDition
STO		STOP

7.1.1.2 Langage machine

Il y a donc neuf types d'instructions primitives dans notre modèle. On peut donc en coder le type sur quatre bits, chaque type d'instruction étant associé (de façon *a priori* arbitraire) à un nombre en binaire de 0 à 15. L'adresse, elle, sera codée sur n bits avec n choisi tel que $2^{n-1} < m \leq 2^n$ si m est le nombre maximum de registres.

Attribuons donc un code à chaque dénomination mnémotechnique, appelé **code opération**, ou **opcode** en abrégé, par exemple :

Mnémonyme	Opcode
ENT	0000
SOR	0001
CHA	0010
CHC	0011
RNG	0100
ZER	0101

UNI	0110
ADD	0111
STO	1000
RDS	1001

7.1.2 Notion de registre d'instruction

Le programme peut être rangé dans une suite de registres d'un nouveau type, appelés **registres d'instruction**, pouvant contenir chacun $n + 4$ bits et notés i_1, i_2, ..., i_p, ... Par exemple si la première instruction est `ENT` e_1 alors le contenu de i_1 sera (en supposant $n = 5$) :

0000 00001.

7.1.3 Décodage des instructions primitives

Considérons le registre d'instruction i_k. Notre problème est de **décoder l'instruction**, c'est-à-dire de faire effectuer l'instruction primitive qui y est contenue.

Abandonnons momentanément notre problème de codage et de décodage des instructions pour étudier un circuit combinatoire qui nous sera utile pour ce propos.

7.1.3.1 Multiplexeur

Notion.- Jusqu'ici nous n'avons utilisé une suite de bits que pour représenter des entiers naturels, en se servant tout naturellement de la numération binaire. Nous venons de voir comment les utiliser pour coder les instructions. Plus généralement une suite de n bits permet de **coder** n'importe quel élément d'un ensemble fini de cardinal inférieur à 2^n.

Cet ensemble fini sera concrétisé pour nous par au plus 2^n fils. On aimerait bien un circuit combinatoire permettant des opérations de codage et de décodage tel que représenté à la figure 7.1.

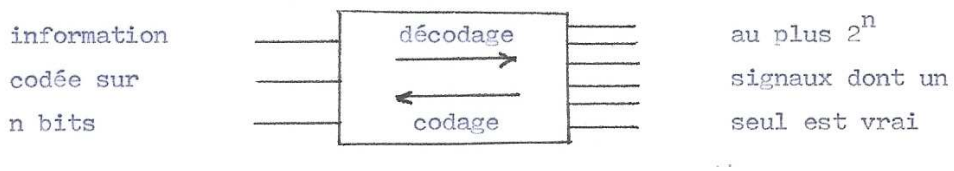


FIGURE 7.1 – Codage et décodage

On appelle **multiplexeur** et **démultiplexeur** tout circuit électronique qui permet de passer de l'un à l'autre.

Réalisation.- Il s'agit de circuits combinatoires, on sait donc les réaliser avec des portes NON, OU et ET.

En pratique on utilise des **matrices de diodes** bien adaptées à ce type de problème.

7.1.3.2 Décodage des instructions primitives par l'ordinateur

Pour décoder une instruction primitive, il suffit d'utiliser un multiplexeur, c'est-à-dire une matrice de décodage, à $n + 4$ entrées, c'est-à-dire la taille des registres d'instruction, et à $3 \times m+6$ sorties, notées `T`, `C`, `Z`, `U`, `A`, `S` pour le type d'instruction et `E1`, ..., `Em`, `S1`, ..., `Sm`, `R1`, ..., `Rm`, pour l'adresse. Si les entrées reçoivent le contenu de i_k alors au plus deux sorties seront positives. Ces sorties sont déterminées par le tableau suivant :

i_k	Sorties	positives
ENT n	T	En
SOR n		Sn
CHA n	T	Rn
CHC n	C	Rn
RNG n		Sn
ZER	Z	
UNI	U	
ADD n	A	En
STO	S	

On comprend alors comment l'instruction primitive est réalisée en tenant compte de ce que nous avons vu précédemment.

7.2 Programmes simples

Nous venons de voir comment on peut décoder *une* instruction (primitive) et la réaliser. Un **programme simple** est une suite d'instructions primitives. Si celles-ci sont placées dans les registres d'instruction, il faut aller les chercher une à une et les faire exécuter. Nous allons voir comment on peut réaliser ceci.

Pour cela nous allons commencer par étudier trois nouveaux circuits séquentiels : le bistable J.K., l'horloge et le registre à incrémentation.

7.2.1 Nouveaux composants électroniques

7.2.1.1 Bistable J.K.

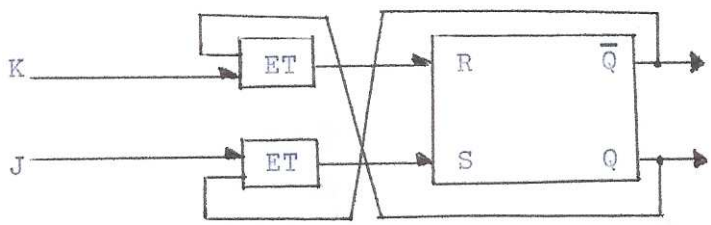


FIGURE 7.2 – Bistable J.K.

Principe.- La figure 7.2 permet de comprendre le fonctionnement d'un **bistable J.K.** : les entrées J et K sont impulsionnelles. La particularité de ce bistable est qu'il bascule systématiquement

d'un état à un autre en cas d'impulsions simultanées sur J et K, sous réserve que les impulsions soient courtes vis-à-vis du temps de basculement.

On a la **table de transition** ci-dessous, en notant Q' l'état précédent et Q l'état après impulsion :

J	K	Q'	Q	Signification de la transition
1	0	0	1	mise à 1
0	1	0	0	reste à 0
1	1	0	1	changement d'état à 1
1	0	1	1	reste à 1
0	1	1	0	mise à zéro
1	1	1	0	changement d'état à zéro

La notation d'un bistable J.K. est représentée sur la figure 7.3.

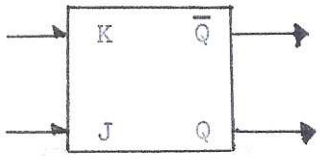


FIGURE 7.3 – Notation d'un bistable J.K.

Application.- Le bistable J.K. permet de réaliser, comme le montre la figure 7.4, un élément mémoire binaire à trois entrées : S pour la mise à 1, R pour la mise à 0 et C pour la complémentation.

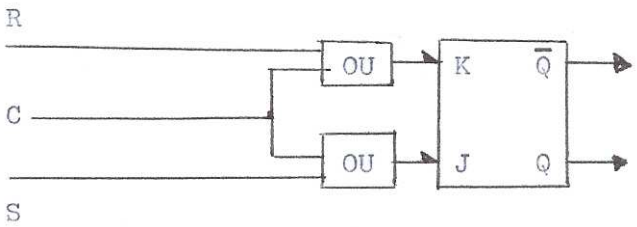


FIGURE 7.4 – Élément de mémoire J.K.

Un élément de mémoire J.K. est noté comme sur la figure 7.5.

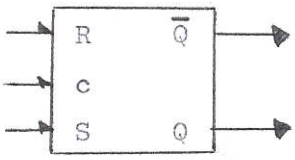


FIGURE 7.5 – Notation d'un élément de mémoire J.K.

7.2.1.2 Bascule et horloge

Principe.- Une **bascule** est un élément de circuit séquentiel n'ayant aucune position stable. On peut en réaliser une en reliant les deux entrées d'un bistable J.K. de la façon indiquée par la figure 7.6 : tant que l'entrée T est maintenue à 1, la bascule change d'état constamment, le rythme de basculement dépendant du temps de réponse des circuits. Ce rythme est ajustable grâce aux temporisateurs θ_1 et θ_2.

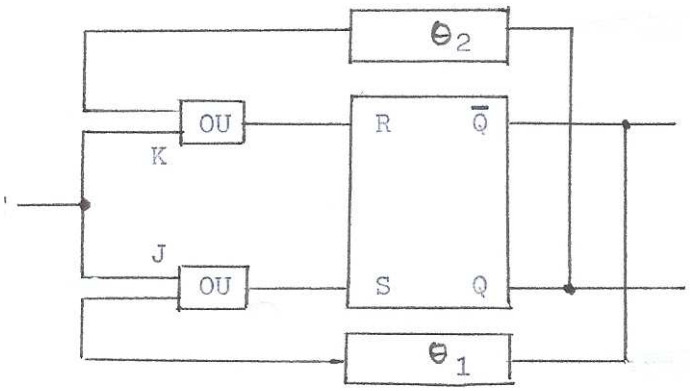


FIGURE 7.6 – Bascule J.K.

Application.- On peut utiliser la bascule pour réaliser une **horloge** fournissant un train d'impulsions régulièrement espacées, conformément à la figure 7.7.

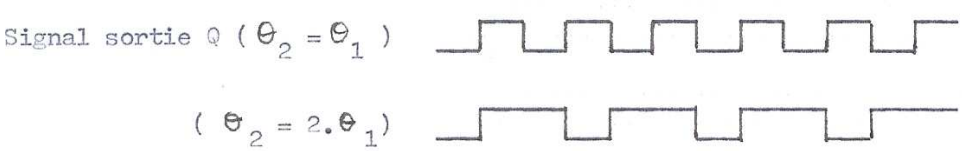


FIGURE 7.7 – Train d'impulsions

7.2.1.3 Registre à incrémentation

Notion.- Un **registre à incrémentation**, ou **compteur d'impulsion**, est un registre dont le compteur s'incrémente de 1 chaque fois qu'une impulsion se présente à l'entrée. Il est donc capable de compter le nombre d'impulsions reçues.

Principe.- Un registre à impulsion peut être réalisé à l'aide de bistables J.K. de la façon montrée à la figure 7.8.

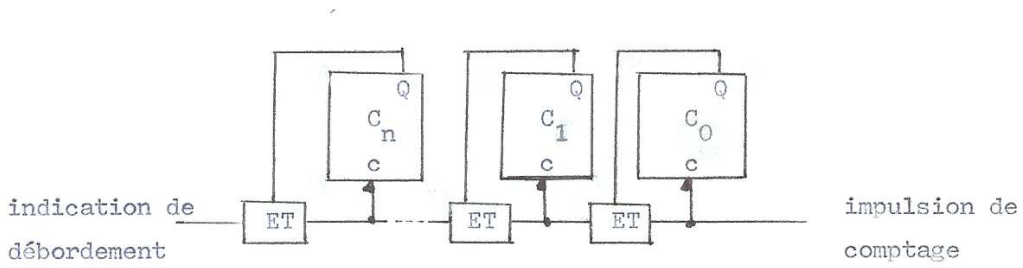


FIGURE 7.8 – Registre d'incrémentation

L'impulsion attaque le bistable de plus faible poids. S'il est à 0, il passe à 1 ; s'il est à 1, il doit passer à 0 mais de plus l'impulsion doit être propagée au bistable suivant pour tenir compte de la retenue.

7.2.2 Exécution du programme

Pour exécuter le programme simple placé dans les registres d'instruction, on se sert d'un compteur de programme.

Compteur de programme.- Supposons que notre calculateur électronique comporte 2^p registres d'instruction. On monte sur ce calculateur un registre à incrémentation (RAI en abrégé) de p bits, appelé **compteur de programme** ou **PC** (pour l'anglais *Program Counter*).

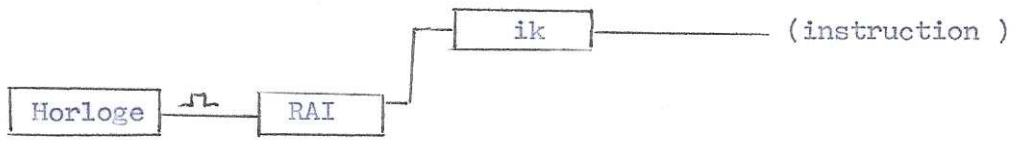


FIGURE 7.9 – Compteur de programme

Cadencement.- On associe le compteur de programme à une horloge qui va lui envoyer régulièrement des impulsions, appelées **tops** d'horloge, comme indiqué à la figure 7.9.

Exécution du programme enregistré.- Le compteur de programme peut être mis à zéro (comme tout registre, de la façon vue précédemment, et ici par exemple en appuyant sur un bouton avant de démarrer l'exécution du programme). Chaque top introduit un entier dans le PC, en commençant donc par 0 et incrémenté de 1 à chaque fois. On interprète cet entier comme une adresse d'instruction k, codée en binaire. Un circuit va chercher cette instruction dans le registre i_k et la décode de la façon vue antérieurement, ce qui permet d'effectuer cette instruction primitive.

Il faut, d'autre part, penser à l'instruction suivante. Puisque le PC est un registre à incrémentation, l'adresse qu'il contient est automatiquement incrémentée de 1, ce qui permet d'avoir l'adresse de l'instruction suivante.

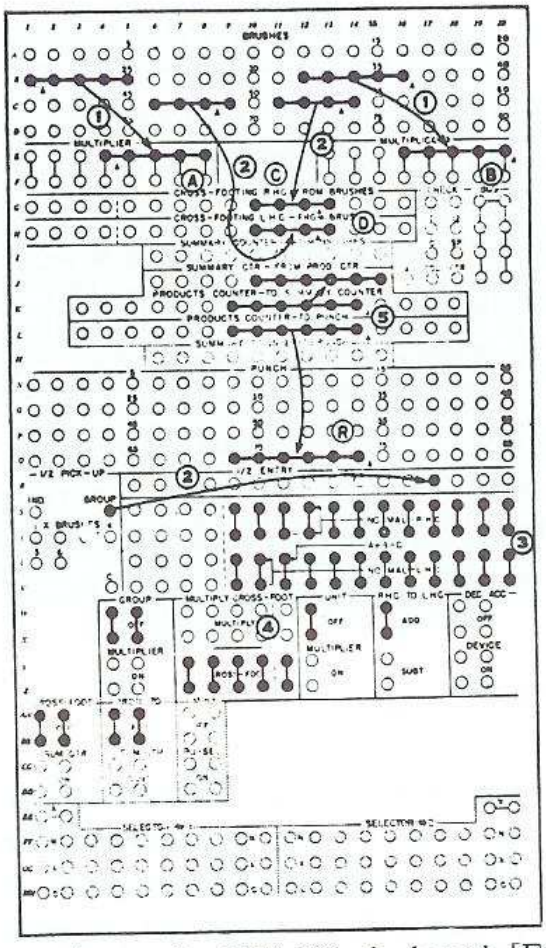
7.2.3 Déroulement d'un programme

On voit comment utiliser la machine que nous venons de décrire :

1. On place le programme dans les registres d'instruction (par exemple des registres à interrupteur).
2. On place les données dans les registres d'entrée (à interrupteurs également, par exemple).
3. On initialise le PC à zéro (grâce à un bouton spécial par exemple, qui peut aussi être déclenché à l'étape 4).
4. On appuie sur un bouton DÉPART (dont le rôle est de connecter l'horloge et donc d'envoyer des tops).
5. Le programme se déroule alors, comme nous l'avons vu, et il arrivera à un certain moment sur une instruction STOP, qui a pour effet de déconnecter l'horloge, d'une part, et d'autre part de faire clignoter une ampoule pour indiquer à l'utilisateur la fin du programme.
6. On peut alors aller chercher les résultats dans les registres de sortie (à ampoules, par exemple).

7.3 Historique

Dans notre exposé, nous sommes parti d'un langage symbolique, traduit en langage machine puis nous sommes passé à la mise en œuvre d'un tel langage sur une machine. Pour l'aspect historique, nous allons faire l'inverse, en commençant par décrire les machines.



Wiring diagram for IBM 601 plugboard. [From IBM (1947).]

FIGURE 7.10 – Programmation par câblage ([Bur-80], p. 328)

7.3.1 Aspect matériel

7.3.1.1 Calculateurs à programmes câblés

On est passé des calculateurs effectuant les quatre opérations (l'analogue des calculettes quatre opérations actuelles) à une programmation (programmation séquentielle) câblée, d'abord

pour des machines à relais électromagnétiques (l'IBM 601) puis pour des machines électroniques (l'ENIAC).

IBM 601 (1935)

La première machine complexe qui encouragea IBM à se lancer sur le marché des calculateurs est l'IBM 601, dite *Multiplying Punch*, introduit en 1935. Il s'agit d'une machine à relais pouvant multiplier deux nombres en à peu près une seconde. Elle connaît un énorme succès commercial à la fin des années 1930 et dans les années 1940, puisque vendue à 1 500 exemplaires.

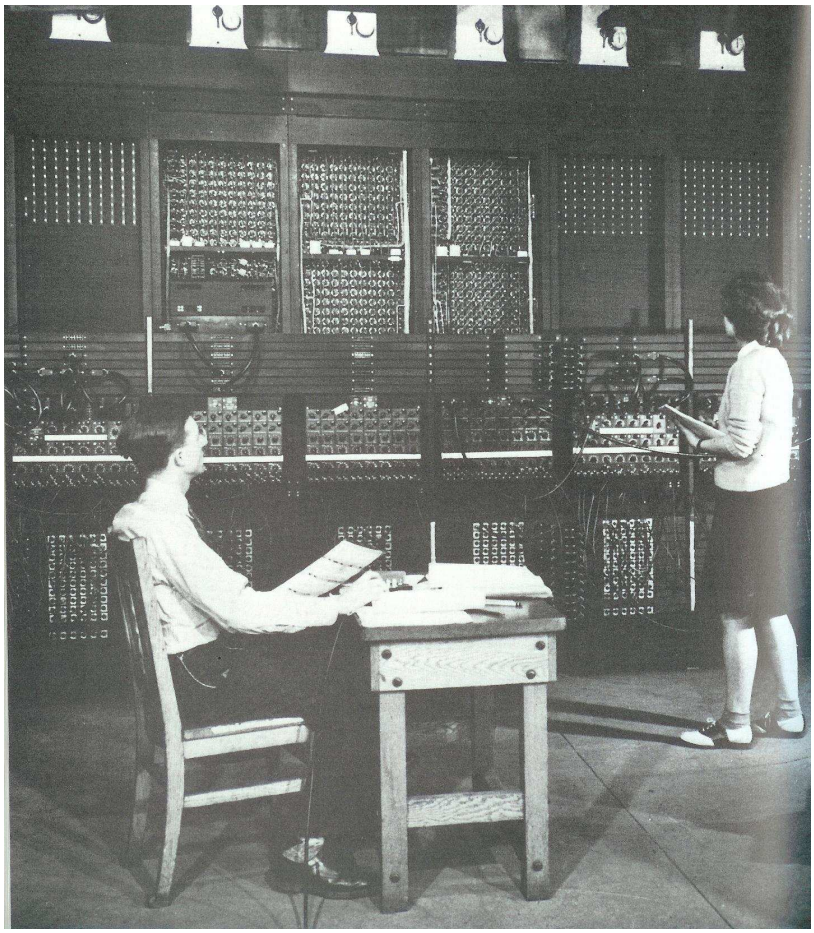


FIGURE 7.11 – Programmation par câblage ([Bur-80], p. 317)

Cette machine se programme par câblage. Le tableau de câblage de la figure 7.10 est prêt pour calculer $A \times B + C + D$:

– Le branchement 1 place les contenus de A et de B dans les compteurs appelés *ier* (pour *multiplIER*) et *icand* (pour *multiplICAND*) respectivement. L'IBM 601 multiplie un chiffre du *ier* par tous les chiffres de *icand* en un temps d'addition et ajoute le résultat à deux compteurs, le compteur de la partie gauche (LHC pour *left-hand counter*) et un pour la partie droite (RHC pour *right-hand counter*).

- Le branchement 2 permet de transférer initialement les contenus de C et de D dans les compteurs RHC et LHC de façon à ce qu'ils apparaissent dans la somme finale. Les positions des broches dépendent de la position du point décimal.
- Le branchement 4 permet d'ajouter les résultats pas à pas de la multiplication dans RHC et LHC.
- Le branchement 4 concerne le contrôle de façon à ce qu'on ajoute C et D à $A \times B$ et que l'on n'effectue pas seulement $A \times B$.

ENIAC (1945)

La programmation de l'ENIAC (1945) est inspirée de celle de l'IBM 601 avec des tables de fonctions et des câbles enfichables. Cette opération initiale peut demander jusqu'à deux journées de travail dans des cas complexes. La programmation ne peut pas être conservée, il faut l'effectuer à nouveau lorsqu'on est passé à un autre programme entre temps.

La photo 7.11 montre le mathématicien Arthur Burks indiquant à une technicienne le câblage à effectuer concernant un de ses programmes sur l'ENIAC.

Deux raisons expliquent l'absence de sophistication des procédures de programmation et de lancement des travaux : en premier lieu, l'impatience des militaires fait écarter tout perfectionnisme risquant de retarder la mise en service du calculateur ; en second lieu, l'ENIAC est réalisé essentiellement comme outil destiné à résoudre des problèmes balistiques, or très peu de modifications sont nécessaires pour calculer une même famille de trajectoires. Dans le rapport sur l'avancement de l'ENIAC du 31 décembre 1944, il est écrit :

> *Aucun essai n'a été entrepris pour initialiser un programme automatiquement. Ceci est dû a un souci de simplicité et puisqu'il est envisagé que l'ENIAC soit principalement utilisé pour des problèmes d'un type pour lequel une initialisation sera utilisée un grand nombre de fois avant qu'un autre problème ne soit placé sur la machine.*
>
> [Ste-81], p. 75

Par contre, dans leur proposition de l'ENIAC de 1943, Mauchly, Eckert et Brainerd mentionnent la possibilité de centraliser le contrôle par des cartes perforées mais ceci ne sera jamais mis en place.

7.3.1.2 Calculateurs commandés par programme externe

Pendant que se développent ces calculateurs à programmes câblés aux États-Unis, en Allemagne Konrad Zuse fait émerger la notion se *programme séquentiel* (sans se rendre compte qu'il peut exister d'autres types de programmes) et met au point des machines permettant d'exécuter ces programmes, externes à la machine.

Nous avons vu au chapitre un comment Konrad Zuse est parvenu à sa notion de programme séquentiel. Il conçoit alors un certain nombre de calculateurs généralistes. La suite d'opérations arithmétiques élémentaires n'a pas besoin d'être prévue au moment de la conception du calculateur. Un *programme* extérieur vient spécifier au calculateur la suite d'opérations à effectuer. Nous avons vu, dans l'extrait cité au chapitre un, un programme pour le calcul d'un déterminant 3×3. Pour le mettre en œuvre, Konrad Zuse s'inspire du procédé mis en place à la fin du XVIII^e siècle par Falcon et Jacquard pour automatiser le tissage des étoffes façonnées.

Automates

Les horlogers, après avoir réglé le problème de l'affichage de l'heure, munissent leurs horloges d'automates à figures humaines à partir du XIII^e siècle. L'âge d'or des automates est le XVIII^e siècle avec, entre autres artisans renommés, Jacques de Vaucanson (1709–1782) en France et les frères Jacquet-Droz, Pierre (1721–1790) et Henri-Louis (1752–1791), en Suisse. Comment

fonctionnent ces automates ? La technique principale qui permet de mettre ces simulacres en mouvement est la technique de l'*arbre à came*, inventée par les Grecs et signalée au II^{e} siècle par Héron d'Alexandrie dans un ouvrage appelé *Pneumatica* ([Héron], section 78).

Qu'est-ce qu'un arbre à came ? Il s'agit d'une roue, d'un cylindre, ou d'une quelconque pièce découpée, taillée de façon à ce que chaque crantage engendre un mouvement différent de l'ensemble du dispositif auquel elle est reliée. L'exemple le plus connu de « cylindre à cames » est celui de la boîte à musique formée d'un rouleau hérissé de picots contre lesquels viennent buter les dents d'une sorte de peigne métallique. Ces « dents », de longueurs inégales, vibrent alors et produisent une note déterminée.

Les automates se distinguent par la finesse de réalisation des différentes pièces et par leur agencement. Mais le cœur du dispositif est la *programmation*, c'est-à-dire le contrôle selon un ordre déterminé, des mouvements de l'ensemble. Le changement du cylindre à cames entraîne l'automate à jouer d'autres pièces, à imiter d'autres comportements.

Perforations

La mise en œuvre de ce principe de programmation, pour un but autre que le calcul, se déplace au XVIII^{e} siècle des dispositifs à cames vers des machines utilisant des *cartes perforées*. Celles-ci sont mises au point entre 1728 et 1734 par Basile Bouchon, puis par Falcon (1705–1765), pour automatiser le tissage des étoffes façonnées. Le mécanicien français Joseph-Marie Jacquard (1752–1834) perfectionne le dispositif en s'inspirant également du métier à tisser que Vaucanson a mis au point précédemment.

Z1

En 1938 Zuse expérimente un additionneur de deux nombres binaires puis met au point un prototype, appelé *Versuchmodell I* (modèle expérimental 1) ou **V1** (renommé **Z1** après 1945). Il est entièrement mécanique et ne fonctionne pas très bien [Zus-80]. Le programme est contrôlé par un ruban perforé.

Z2

Un étudiant en télécommunication au collège technique de Berlin, Helmut Schreyer, suggère à Zuse d'utiliser de vieilles pellicules de film cinéma 35 millimètres pour y perforer les données et les instructions, ce support étant moins cher et plus solide que les rubans de papier, et de remplacer la partie mécanique par des relais téléphoniques d'occasion pour l'unité arithmétique tout en conservant une mémoire mécanique. Confiant dans sa conception, c'est ce que fait Zuse en 1939, qui baptise l'ensemble *Versuchmodell 2*, ou **V2** (renommé **Z2** après 1945).

Zuse décide de présenter ce prototype aux autorités influentes de l'aéronautique allemande. Mais avant qu'une telle démonstration ne puisse être organisée, l'Allemagne envahit la Pologne et se retrouve en guerre ; malgré plusieurs demandes d'affectation spéciale, Zuse, qui a alors 29 ans, est rappelé comme soldat puis démobilisé six mois plus tard. Il est alors affecté aux bureaux d'études du constructeur aéronautique Henschel, où il termine son Z2 à ses heures perdues.

Z3

Zuse arrive à joindre Alfred Teichmann, l'un des directeurs de l'Institut allemand de recherche aéronautique, et l'invite à venir, chez ses parents, voir fonctionner le calculateur V2. Cette démonstration convainc Teichmann qu'il faut aider Zuse à construire un modèle définitif, ce qui est concrétisé par quelques fonds pour acheter des composants électromagnétiques, mais sans fournir atelier ou collaborateur.

Parce qu'il est impossible de construire une nouvelle machine dans l'appartement déjà encombré de ses parents, Zuse loue un logement au n^{o} 6 d'Oranienstrasse. Sa **V3** (renommée **Z3** après 1945) y est achevée le 5 décembre 1941 et reste installée dans la cave jusqu'à ce qu'un bombardement aérien détruise l'immeuble, le 6 avril 1945. Il n'existe ni vestige ni photo du troisième calculateur de Zuse. Tout ce que l'on en sait repose sur les diagrammes joints aux demandes de brevets déposés en 1941. Zuse indique qu'il existe des différences sensibles entre les plans soumis et le modèle effectivement fabriqué ; le *Deutsches Museum* (Musée allemand des techniques) de Munich a pu faire construire, en 1963, une réplique de ce calculateur binaire, à relais et à programme externe sur ruban. Il coûta environ 25 000 Reichsmark (soit 6 500 dollars

de 1941) en matériel.

C'est une petite machine consistant en un lecteur de ruban perforé, une console opérateur et deux armoires contenant 2 600 relais. Sa mémoire stocke 64 nombres de 22 bits. La multiplication de deux nombres s'effectue en trois à cinq secondes. Outre les quatre opérations arithmétiques, le calcul de la racine carrée est implémentée. Les valeurs initiales sont entrées à la main et le programme est guidé par le ruban perforé. Le programme, ou *Rechenplan*, énumère toutes les opérations arithmétiques à effectuer, plus les opérations annexes (sauvegarde des résultats intermédiaires et leur rappel de la mémoire lors de leur utilisation ultérieure). Le plan préparé est exécuté par l'unité de contrôle du calculateur, grâce aux perforations du film-programme. Lorsque la séquence prévue est achevée, une commande « fin » arrête la machine. L'opérateur frappe alors la touche de conversion de binaire en décimal pour provoquer l'affichage du résultat sur les voyants du panneau de la console. Des voyants lumineux particuliers signalent les anomalies de calcul décelées par la machine (dépassement de capacité, division par zéro, résultats impossibles, résultats indéterminés).

S1 Bien que le Z3 fonctionne bien, l'Institut préfère des machines dédiées, aussi Zuse construit-il un premier calculateur dédié, appelé **S1** pour analyser les vibrations d'ailes des bombes planantes Henschel 293. Celui-ci est satisfaisant et les trente femmes alors affectées à ce problème sont transférées à une autre fonction. Le S1 est opérationnel de 1942 à 1944, jusqu'à ce qu'une bombe alliée mette un terme à son existence.

S2 Entre-temps, Zuse a conçu un deuxième modèle de machine dédiée, plus perfectionné, capable de prendre en compte lui-même les signaux électriques transmis par des appareils de mesure intégrés au prototype de bombe en cours d'expérimentation. Cette **S2** est pratiquement achevée lorsque le transfert de l'usine Henschel hors de Berlin est décidé. Tandis que la S2, démontée, attend son transport, le local où elle se trouve entreposée est rasé par une attaque aérienne.

Z4 Encouragé par ce succès, Zuse s'embarque dès 1942 dans une version qu'il considère comme son modèle définitif, le **V4** (renommé **Z4** après 1945), comportant une mémoire de 512 nombres de 32 bits, taille indispensable à l'exécution de grands calculs. Une construction au moyen de relais conduisant à des dimensions inacceptables (29 mètres de longueur, 2 mètres de haut, 30 centimètres de profondeur), Zuse reprend alors le principe de la mémoire mécanique.

Pendant la construction, l'atelier de Zuse est endommagé plusieurs fois par les bombardements alliés de 1944 et il faut, à trois reprises, déménager la machine dans divers quartiers de Berlin. À partir des premiers mois de 1945, les attaques aériennes sur Berlin devenant quotidiennes, les autorités décident l'évacuation de la V4. La machine est d'abord transférée dans la petite ville universitaire de Göttingen, à 160 kilomètres à l'ouest de Berlin. Là, dans les locaux de l'institut d'aérodynamique expérimentale, on procède aux derniers montages et aux essais.

Après avoir encerclé la Ruhr, les forces américaines se dirigent vers l'est et leurs avant-gardes atteignent Kassel. On démonte la V4 et il est décider de cacher ses pièces dans les fortifications souterraines du massif du Harz qui, grâce à un relief abrupt, doit constituer un centre de résistance puissant. Sur les documents de transport la concernant, la machine est identifiée sous le sigle V4. Les agents du ministère des Transports croient qu'il s'agit du prototype d'une nouvelle arme secrète et, malgré la pénurie de véhicules et d'essence, fournissent à Zuse tous les moyens nécessaires et l'ordre de gagner, au plus vite, le « réduit bavarois » qui doit constituer la dernière forteresse des nazis. Par Hof, Munich, Ettel et Hindelang, le petit convoi, voyageant de nuit, tous phares éteints, parvient jusqu'à l'Allgäu, province située à proximité de la frontière autrichienne et Zuse échoue dans le minuscule village d'Hopferau.

Bientôt les troupes de la 1^{ere} armée française occupent la région ; le village est fouillé sans que la machine, cachée dans la cave d'une ferme, soit découverte. Lorsque la délimitation des zones d'occupation est entérinée, les Français laissent, dans ce secteur, la place aux Américains. Après la reddition allemande, Zuse tente de constituer une petite équipe pour assembler la

V4 dans la grange d'une ferme. L'existence de cette machine parvient à la connaissance des forces d'occupation américaines. Le capitaine Robert E. Work, de l'armée de l'air, dépendant du Q.G. des forces américaines en Autriche, rédige un rapport sur cette « machine à calculer automatique », rendu public le 8 novembre 1946. Un autre officier, Roger C. Lyndon, voit également la machine et rédige un rapport en 1947 ([Lyn-47]).

La Suisse

Edouard Stiefel (1908–1978), de l'École Polytechnique de Zurich, fait, en 1948, un voyage aux États-Unis. Il voit les Mark I et II d'Harvard ainsi que l'ENIAC. Il mesure tout ce que les calculateurs automatiques peuvent apporter et cherche comment mettre une machine de ce type à la disposition de l'École Polytechnique. C'est seulement à son retour en Europe que Stiefel entend parler de la machine de Zuse. Stiefel et Zuse se rencontrent à Hopferau, le 13 juillet 1949. Stiefel remet une équation différentielle à Zuse ; celui-ci la programme devant lui et la machine la résout immédiatement. Stiefel rentre à Zurich et établit une proposition de location de la V4 pour cinq ans, temps jugé nécessaire pour construire un calculateur électronique entièrement suisse. Après de longs pourparlers, le texte définitif du contrat est établi : l'école loue la machine pour cinq ans ; Zuse garantit son bon fonctionnement pendant cette période ; il accepte le prix proposé de 30 000 francs suisses, mais exige un paiement unique, à l'ouverture de la location. C'est à ce moment que la machine est rebaptisée : la *Versuchmodell* 4 devient la « Zuse 4 ». Depuis, toutes les machines de Zuse sont identifiées par la lettre Z suivie de leur numéro d'ordre. Le contrat est signé fin 1949 : Zuse, citoyen allemenand, n'est pas autorisé à quitter le territoire occupé ; les Suisses n'acceptent pas de signer le contrat hors de leur territoire national ; un terrain neutre est trouvé : le buffet de la gare internationale de Bâle.

Zuse peut alors créer une petite société, « Zuse KG », ayant son siège à Neukirchen. Celle-ci est rachetée par Siemens au début des années 1960.

La Z4, après révision générale, est livrée à l'École Polytechnique de Zurich le 11 juillet 1950. La machine y reste opérationnelle jusqu'en 1954. Lorsque le calculateur suisse Ermeth est en cours d'achèvement à l'École polytechnique, la Z4 est transférée à l'institut franco-allemand de la recherche, à Bâle, où elle travaille pendant une nouvelle période de cinq ans.

7.3.1.3 Programmes séquentiels en mémoire électronique

Babbage et Zuse

On a dit que Babbage avait conçu un calculateur à programme enregistré en se référant à un bref passage de la note A de la traduction par Lady Lovelace de l'article de Menabrea. Ce passage fait référence, de façon plutôt obscure, à des nombres représentant des opérations plutôt que des données. Les études de Wilkes [Wil-71] de quelques-uns des carnets de notes de Babbage l'ont amené à la conclusion qu'il n'avait pas une idée très claire de la notion de programme.

Nous avons vu ci-dessus que l'article de 1936 de Zuse contient une référence claire à la possibilité d'enregistrer les programmes mais il ne semble pas avoir développer cette idée.

Première idée : l'EDVAC (conçu en 1945)

Les programmes (séquentiels) externes représentent un progrès énorme par rapport à la programmation par câblage mais, puisque les besoins se montrent de plus en plus exigeants, une contrainte nouvelle apparaît bientôt : le décalage de vitesse entre l'unité arithmétique et l'accès aux périphériques sur lesquels se trouve le programme externe. Il faut donc faire passer le programme des périphériques en mémoire interne de façon à y accéder à la vitesse d'exécution des calculs.

Nous avons vu que le premier calculateur électronique vraiment utilisé est l'ENIAC, conçu à la *Moore School* par ECKERT et MAUCHLY. Avant même que l'ENIAC soit terminé, en juin 1944, ECKERT et MAUCHLY pensent à un calculateur à programme (séquentiel) enregistré, avec une mémoire contenant à la fois les données et le programme, ceci certainement à la fin 1943.

Herman GOLDSTINE demande au BRL de sponsoriser le développement d'un autre calculateur. En octobre 1944, l'*Ordnance Department* passe un contrat de 105 000 $ pour la conception de l'EDVAC (*Electronic Discrete VAriable Computer*).

Le problème avec l'ENIAC est qu'il a une capacité de stockage très réduite. Il faut donc, dans une première étape, améliorer la technologie des mémoires. Curieusement, si les mémoires actuelles reposent sur les mêmes principes que la mémoire de l'ENIAC (c'est-à-dire sur des bascules comme élément primitif), les technologies développées dans les années 1944 à 1946 sont différentes.

Perry CRAWFORD propose une mémoire à disque magnétique et Pres ECKERT, se rendant compte qu'un calculateur avec une telle mémoire serait supérieur à l'ENIAC, propose un telle machine au début de 1944 [Eck-44]. Plus tard il écrit :

> *En janvier 1944, j'écrivis un mémo, Divulgation d'une machine à calculer magnétique, que j'ai tapé sur ma machine à écrire à la maison et que je donnais à mon directeur de thèse pour qu'il soit retapé. Pour des raisons inconnues il ne fut pas retapé mais j'ai finalement retrouvé ma propre version. J'ai aussi lu un mémoire de master de Perry Crawford, du MIT, où il avait proposé d'utiliser un disque avec des points magnétisés pour le stockage des nombres. Mon mémo disait que nous pourrions utiliser des disques magnétiques effaçables ou permanents pour le stockage de l'information à la fois altérable et non changeable. Le concept de stockage interne général est parti de ce mémo.*
>
> [...]
>
> *Ma meilleure idée sur le calcul, aujourd'hui appelée brièvement « programme stocké », devint pour nous une « idée naturelle ». Il était évident que les instructions du calculateur dévaient être sous la forme d'un code numérique.*
>
> [Eck-80], pp. 530–531

La dernière phrase n'est pas vraiment plus explicitée que cela.

Peu après ECKERT et MAUCHLY proposent le développement de mémoires électroniques utilisant des tubes à lignes de retard, s'inspirant de la technologie utilisée pour les radars. Un travail préliminaire est réalisé par Pres ECKERT et Kite SHARPLESS. Les essais sont concluants et montrent qu'avec un tel tube on peut détenir 100 fois plus de bits qu'avec une bascule.

Il est alors décidé de construire un calculateur à programme (séquentiel) enregistré, l'EDVAC, de capacité mémoire de 1 024 mots de 32 bits.

Les plans de l'EDVAC prévoient que chaque instruction ait la forme d'un code numérique (par exemple 101 1001 1111 0011). Le premier nombre (101) indique l'opération qui doit être effectuée (101 signifie additionner). Les deux nombres suivants (1001 et 1111) indiquent l'adresse en mémoire des quantités sur lesquelles portent l'opération (à l'adresse 1001 il y a par exemple l'entier 3, à l'adresse 1111, l'entier 5). Le troisième nombre (0011) donne l'adresse en mémoire où la machine ira stocker le résultat. On voit l'analogie avec la conception de ZUSE.

La première documentation écrite est le rapport de John VON NEUMANN [Von-45] sur lequel nous reviendrons dans la partie logicielle.

Le programme prend du retard et l'EDVAC (sous une forme modifiée) ne sera opérationnel qu'en 1951, trois ans après que les Anglais auront fait fonctionner le premier ordinateur. En effet, ECKERT et MAUCHLY pensent qu'une telle machine a un réel avenir *commercial*. Il faut

donc, selon eux, déposer un brevet et constituer une société privée dans le but de faire des bénéfices. Les autres membres de l'équipe désapprouvent fortement cette ambition. En 1947, après pratiquement deux ans de querelles intestines qui paralysent la construction de l'EDVAC, la justice conclut que les principes de base de l'ordinateur tels qu'ils sont consignés dans les plans de l'EDVAC appartiennent désormais au domaine public et ne peuvent donc pas faire l'objet d'une prise de brevet.

Première description fonctionnelle : l'IAS (conçu en 1946)

Si le rapport de VON NEUMANN [Von-45] est considéré comme le premier document écrit faisant référence à un programme enregistré, il ne fait aucunement mention de la façon de le mettre en place. Celle-ci est abordée pour la première fois à propos de l'IAS.

Quittant le projet de l'EDVAC, VON NEUMANN emmène avec lui GOLDSTINE à l'IAS (*Institute of Advanced Study*, Princeton University) pour concevoir un nouvel ordinateur. Celui-ci s'appelle tout simplement **machine IAS**.

Il trouve un financement auprès de l'*U.S. Army Ordnance Department* et de RCA (*Radio Corporation of America*), qui a un nouveau centre de recherche à Princeton et veut se lancer dans la fabrication des lampes pour les calculateurs.

VON NEUMANN doit tout d'abord convaincre ses collègues de l'IAS de la pertinence de son projet. Il y a une certaine résistance à ce que la technologie entre dans une université prestigieuse, qui se consacre traditionnellement à la seule recherche fondamentale et qui, en outre, cultive un certain élitisme (les étudiants déjeunent en toge au restaurant de l'université). Construire une machine implique la présence d'un cortège d'ingénieurs, de fonds militaires, de contrats, en bref une sorte de compromission avec le domaine des sciences appliquées. Mais VON NEUMANN sait être persuasif. Ne construit-il pas, après tout, un modèle réduit du cerveau ?

Les plans de l'IAS sont décrits sous le titre *Discussions préliminaires sur la conception logique d'un outil de calcul électronique* [Bur-46]. Cet article, paru en 1946, est écrit avec GOLDSTINE et Arthur BURKS, mathématicien à la *Moore School* qui a travaillé sur l'ENIAC et sur l'EDVAC.

Norbert WIENER (1894–1964) recommande Julian BIGELOW (avec qui il a écrit en 1942 un des articles fondateurs de la cybernétique) à VON NEUMANN comme ingénieur en chef du projet.

La machine doit utiliser une mémoire à base de tubes électrostatiques, ou *iconoscope*, plus rapide que les lignes à retard de mercure, avec un accès direct et non plus cyclique, les bits pouvant être lus en parallèle et non plus en série. Cependant cette technologie ne sera jamais vraiment maîtrisée.

Les différents rapports intermédiaires qui circulent jusqu'en 1952, date à laquelle la machine est terminée, inspirent de nombreux projets de machines binaires parallèles, en particulier l'IBM 701, l'**ILLIAC** (*ILLInois Automatic Computer*) à l'université de l'Illinois à Urbana, le **JOHNIAC** (faisant référence au prénom de VON NEUMANN) construit par la Rand Corporation à Santa Monica en Californie et le **MANIAC** (*Mathematical Analyser, Numerator, Integrator, And Computer*) à Los Alamos.

Dans le rapport de 1946 apparaît la première description de la mise en œuvre des programmes enregistrés :

> *1. Principaux composants de la machine*
>
> [...]
>
> *1.2 Il est évident que la machine doit être capable de stocker d'une certaine façon les informations numériques nécessaires à un calcul donné telles que les valeurs limites, les tables de fonctions (telle que l'équation d'état d'un fluide) et aussi les résultats intermédiaires du calcul (qui peuvent être attendus à différents moments), mais aussi*

les instructions qui gouvernent la routine en cours à exécuter sur les données numériques. Dans une machine dédiée, ces instructions sont une partie intégrante de l'outil et constituent une partie de sa conception. Pour une machine versatile, il doit être possible de donner des instructions à l'outil sur le calcul. Il doit donc y avoir un organe capable de stocker ces ordres. Il doit, de plus, y avoir une unité qui peut comprendre ces instructions et ordonner leur exécution.

1.3 Conceptuellement nous avons discuté ci-dessus de deux formes différentes de mémoire : le stockage des nombres et le stockage des ordres. Si, cependant, les ordres donnés à la machine sont réduits à un code numérique et si la machine peut distinguer d'une certaine façon un nombre d'un ordre, l'organe de mémoire peut être utilisé à la fois pour les nombres et pour les ordres. Le codage des ordres sous forme numérique est discuté en 6.3 ci-dessous.

1.4 Si la mémoire pour les ordres est simplement un organe de stockage, il doit exister un organe qui peut automatiquement exécuter les ordres stockés en mémoire. Nous appellerons cet organe le contrôle.

[...]

2. Premières remarques sur la mémoire

[Analysant le problème de la résolution numérique des équations aux dérivées partielles, ils en arrivent à la conclusion :] *En conséquence nous devons prévoir un stockage électronique entièrement automatique d'à peu près 4 000 nombres de 40 chiffres binaires chacun.*

[...]

4. L'organe de mémoire

4.1 [...]

Les formes les plus communes de stockage par des circuits électriques sont la bascule, les tubes à gaz et les relais électro-mécaniques. Réaliser une mémoire de n mots requiert, bien sûr, 40n tels éléments, en excluant les éléments de transfert. Nous avons vu ci-dessus (cf. 2.2) qu'une mémoire rapide de plusieurs milliers de mots n'est pas du tout irraisonnable pour un instrument versatile. Ainsi environ 10^5 *bascules ou éléments analogues seraient nécessaires ! Ceci est bien sûr entièrement irréalisable.*

Nous devons donc chercher une méthode autre que celles suggérées ci-dessus pour stocker les informations électriquement.

[...]

Un outil possèdant cette propriété à un degré marqué est le tube iconoscope. Sous sa forme classique, il possède une résolution linéaire de 500. Ceci pourrait correspondre à une capacité mémoire (bidimensionnelle) de $500 \times 500 = 2.5 \times 10^5$.

[...]

Actuellement les laboratoires de Princeton de Radio Corporation of America sont engagés dans le développement d'un tube de stockage, le selectron, *du type que nous avons mentionné ci-dessus.*

[...]

4.9 Pour les discussions qui suivent, nous supposerons qu'est associé une bascule à la sortie de chaque selectron. Nous appelerons registre de selectron *cet assemblage de 40 bascules.*

5. L'organe arithmétique

[...]

5.2 Dans une discussion des organes arithmétiques d'une machine à claculer, on est naturellement conduit à la considération du système numérique à adopter. En dépit

d'une longue tradition de construction de machines numériques utilisant le système décimal, nous sommes fortement en faveur du système binaire pour notre machine.

[...]

6. Le contrôle

6.1 Il a déjà été dit que le calculateur contiendra un organe, appelé le contrôle, qui peut exécuter automatiquement les ordres stockés dans les selectrons.

[...]

Parmi ces ordres nous pouvons immédiatement décrire deux types majeurs : un ordre du premier type commence par causer le transfert du nombre, qui est stocké à un emplacement mémoire spécifié, des selectrons au registre de selectron. Ensuite, il demande à l'unité arithmétique d'exécuter les opérations arithmétiques sur ce nombre (en général en utilisant un autre nombre qui est déjà dans l'unité arithmétique) et de déposer le nombre en résultant dans l'unité arithmétique. Le second type d'ordres permet le transfert du nombre, qui est détenu dans l'unité arithmétique, dans le registre de selectron, et de là à l'emplacement spécifié dans les selectrons (il se peut aussi que la dernière opération permette un transfert direct de l'unité arithmétique dans les selectrons). Un type d'ordres additionnel consiste des ordres de transfert de 3.5. De plus des ordres contrôlent les entrées et les sorties de la machine.

[...]

6.2 Il est clair, à partir de ce qui vient d'être dit, que le contrôle doit avoir un moyen d'aller à un emplacement spécifié de la mémoire selectron, pour en retirer un nombre pour le calcul ou un ordre. Puisque la mémoire selectron (telle qu'elle est envisagée) contiendra $2^{12} = 4\ 096$ mots de quarante chiffres (un mot étant soit un nombre, soit une paire d'ordres), un nombre binaire de douze chiffres suffit à identifier l'emplacement mémoire. Ainsi un mécanisme est requis qui, recevant un nombre de douze chiffres, sélectionnera l'emplacement mémoire correspondant.

Le type de circuit que nous nous proposons d'utiliser à ce propos est connu sous le nom de table de décodage ou table de fonction plusieurs-un. Il a été développé de différentes façons indépendamment par J. Rajchman et P. Crawford[1]*. Elle consiste en n bascules qui enregistrent un nombre binaire à n chiffres. Elle a aussi un maximum de 2^n fils de sortie. Les bascules activent une matrice dans laquelle les interconnexions entre fils d'entrée et de sortie sont faits de telle façon qu'un et seulement un des 2^n fils de sortie soit sélectionné (i.e. a un voltage positif qui lui est appliqué). Ces interconnexions peuvent être établies au moyen de résistances ou au moyen d'éléments non linéaires (tels que les diodes ou les rectificateurs) ; toutes ces méthodes sont en train d'être analysées.*

[Bur-46]

1. *La table de Rajchman est décrite dans le rapport des laboratoires RCA de Rajchman, Snyder et Rudnick paru en 1943 comme contrat OSRD OEM-sr-591. Le travail de Crawford est discuté dans sa thèse de master au Massachusettes Institute of Technology.*

Première description matérielle : l'EDSAC (1949)

La façon de réaliser matériellement la mise en place du décodage des instructions placées en mémoire électronique est décrite par Maurice Wilkes dans une conférence tenue en juin 1949 à l'université de Cambridge lors de l'inauguration du premier odinateur, l'EDSAC :

Introduction

L'EDSAC (electronic delay storage automatic calculator) est une machine à calculer électronique sérielle travaillant en base deux et utilisant des conteneurs[2] *à ultrasons pour le stockage. La mémoire*[3] *principale consiste en 32 conteneurs, chacun d'à peu près 5 pieds de long et contenant 32 nombres de 17 chiffres binaires, l'un étant le chiffre de signe. Ceci donne 1 024 emplacements de stockage en tout. Il est possible d'utiliser deux emplacements de stockage adjacents ensemble de façon à obtenir un nombre de 35 chiffres binaires (incluant un chiffre de signe) ; ainsi, à tout moment, la mémoire peut contenir un mélange de nombres* longs *et* courts. *Des conteneurs courts, pouvant contenir un seul nombre seulement, sont utilisés comme accumulateur et comme registres multiplieurs dans l'unité arithmétique ainsi que pour le contrôle dans diverses parties de la machine.*

Un code à adresse unique est utilisé dans l'EDSAC, les ordres ayant la même longueur que les nombres courts. Le jeu complet des ordres est le suivant :

Jeu d'ordres de l'EDSAC

A n *Ajouter le nombre de l'emplacement mémoire n à l'accumulateur.*

[...]

T n *Transférer le contenu de l'accumulateur à l'emplacement mémoire n et effacer l'accumulateur.*

[...]

R 2^{n-2} *Déplacer le nombre dans l'accumulateur de n places à droite,* i.e. *le multiplier par* 2^{-n}.

[...]

I n *Lire la ligne suivante de trous sur le ruban et placer les 5 chiffres en résultant dans les positions de poids faible de l'emplacement mémoire n.*

O n *Imprimer le caractère en attente sur le téléscripteur et mettre en attente sur le téléscripteur le caractère représenté par les cinq chiffres de poids fort de l'emplacement mémoire n.*

[...]

Z *Arrêter la machine et faire sonner la clochette d'avertissement.*

Nous verrons que la plupart des ordres consiste en une partie fonctionnelle définissant l'opération et une partie numérique définissant l'emplacement mémoire ; quelques ordres, cependant, consistent uniquement en une partie fonctionnelle.

Un ruban perforé à cinq perforations ordinaire du type de ceux utilisés en télégraphie est utilisé pour les entrées. Chaque ligne de perforations représente un nombre binaire à cinq chiffres et l'opération d'entrée de base consiste à transférer ce nombre en mémoire. De même le mécanisme de sortie est un téléscripteur et l'opération de sortie de base consiste à transférer un nombre binaire de 5 chiffres au téléscripteur

2. *Tank* dans le texte originel.
3. *Store* dans le texte originel.

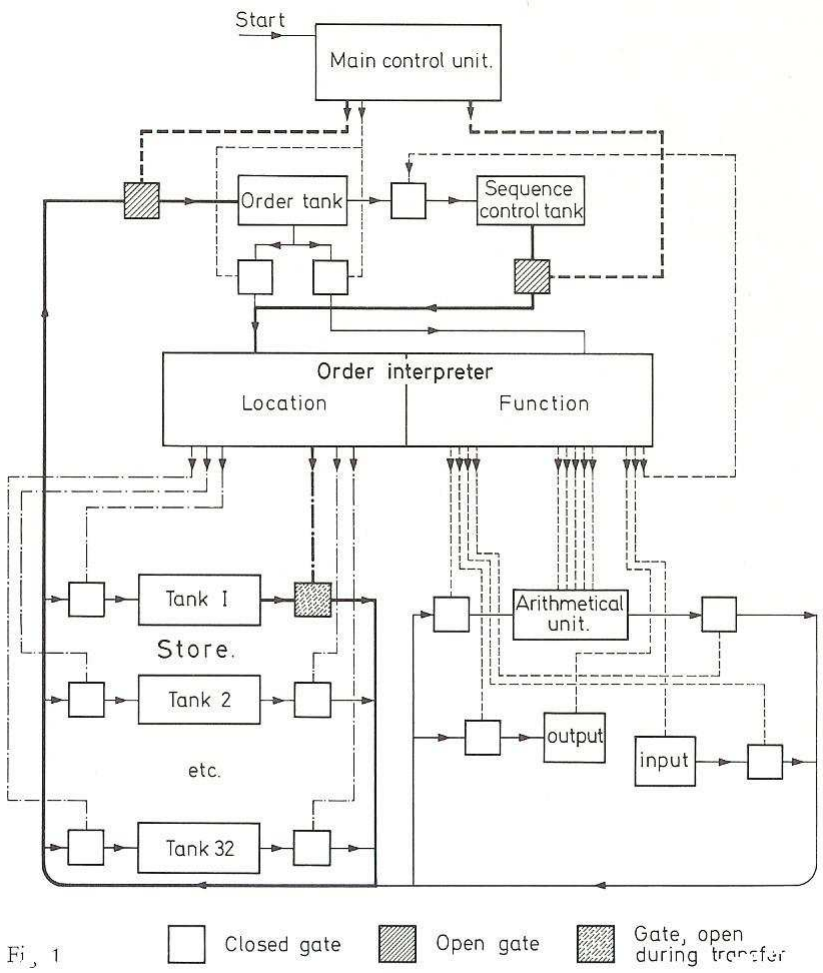


FIGURE 7.12 – ([WR-50], p. 419)

et d'imprimer le caractère correspondant. Le code du téléscripteur est choisi de façon à ce que les nombres binaires jusqu'à neuf soient imprimés comme chiffres décimaux correspondants et un code similaire est utilisé pour l'entrée. Ceci permet à l'opération de conversion du et vers le système décimal d'être programmée comme partie du calcul.

[...]

La suite de contrôle

[...]

Un compteur conserve les ordres dans l'ordre dans lequel ils doivent être exécutés, au moyen d'un conteneur court – connu comme conteneur de contrôle de séquence – auquel est associé un circuit d'addition au moyen duquel une unité est ajoutée au nombre stocké dans le conteneur chaque fois qu'un ordre est exécuté. [...]

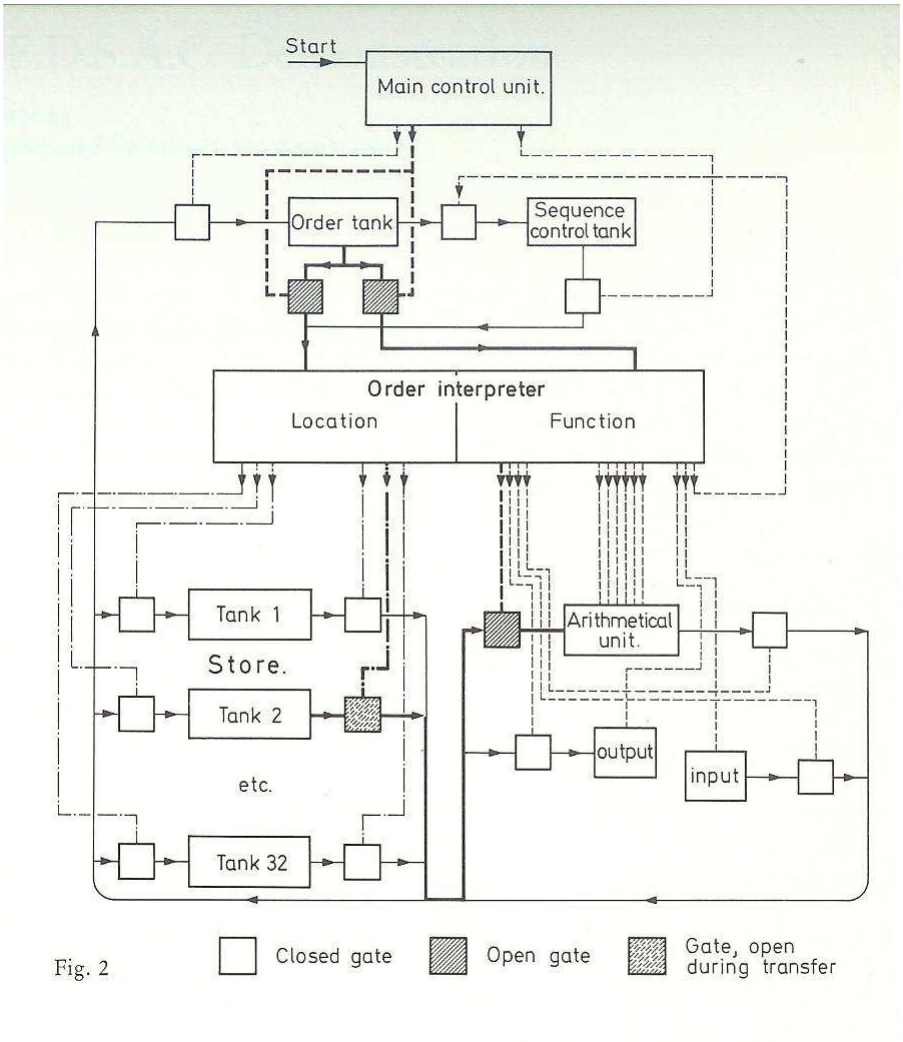


FIGURE 7.13 – ([WR-50], p. 421)

La suite de contrôle de la machine est constituée de deux parties. À l'étape I un ordre est transféré depuis l'emplacement mémoire donné par le nombre du conteneur de contrôle de séquence vers un conteneur court appelé conteneur d'ordre. À l'étape II l'ordre du conteneur d'ordre est exécuté.

Les diverses unités constituant la machine (mémoire, unité arithmétique, unité d'entrée, unité de sortie, conteneur de contrôle de séquence et conteneur d'ordre) sont reliées ensemble à travers des portes de façon à ce que les interconnexions propres à chaque partie successive de la séquence de contrôle puissent être faites en ouvrant les portes appropriées. Les portes sont contrôlées par des impulsions[4] *engendrées soit par l'unité de contrôle principale, soit par une partie de machine donnée, appelée « interpréteur d'ordre » pour cette description. Dans la machine elle-même l'« interpréteur d'ordre » consiste en un certain nombre d'unités séparées. Dans le diagramme*

4. *Waveforms* dans le texte originel.

qui suit, les fils qui transmettent les données[5] *sont représentés par des lignes continues alors que ceux qui transmettent les impulsions le sont en tiretet.*

La figure 7.12 montre l'état de la machine durant la première étape de la suite de contrôle, en supposant que l'emplacement mémoire spécifié par le nombre dans le conteneur de la suite de contrôle soit dans le conteneur 1 de la mémoire principale. Le fil à travers lequel l'ordre passe de l'emplacement de mémoire au conteneur d'ordre est représenté par un trait fort dans le diagramme, et de même les fils à travers lesquels passent les impulsions de contrôle le sont en trait fort tireté. La section d'emplacement de l'« interpréteur d'ordre » reçoit le nombre du conteneur de contrôle de suite et répond en émettant une impulsion qui ouvre la porte de sortie du conteneur 1. Il y a 32 ordres (ou nombres courts, ou un mélange d'ordres, nombres courts et nombres longs) circulant dans le conteneur, et cette porte est ouverte dès que le nombre requis devient disponible et est fermée lorsqu'il l'a traversé. En ceci, elle est traitée différemment des autres portes de la machine, qui restent ouvertes ou fermées durant toute la première étape ; la différence est indiquée dans le diagramme par les différentes formes de traits utilisées. Les interruptions (switching) *et la synchronisation* (timing) *utilisées pour les opérations détaillées ci-dessus sont engendrées par l'« interpréteur d'ordre ».*

L'étape II est très semblable à l'étape I à part le fait que le type d'opération effectuée n'est pas restreint à un simple transfert de la mémoire au conteneur d'ordre. Durant l'étape II, le conteneur d'ordre est relié à l'« interpréteur d'ordre » à la place du conteneur de contrôle de suite et la partie numérique de l'ordre est traitée de la même façon que le nombre du conteneur de contrôle de suite durant l'étape I. Une nouvelle caractéristique, cependant, est que la section de fonction de l'« interpréteur d'ordre » passe à l'action et interprète la partie fonctionnelle de l'ordre ; elle procède en émettant des impulsions qui (a) établissent les connexions entre les unités prêtes pour le passage des nombres et (b) place l'unité arithmétique (ou les unités d'entrée ou de sortie suivant le cas) dans un état de lisibilité pour exécuter l'ordre. La figure 7.13 montre les portes établies pour l'exécution d'un ordre d'addition appelant le transfert d'un nombre du conteneur 2 de la mémoire principale à l'unité arithmétique. Un des fils de contrôle passant par l'« interpréteur d'ordre » à l'unité arithmétique est dessinée comme ligne forte tiretée pour indiquer que l'unité arithmétique a été préparée pour une opération telle que l'addition.

La description ci-dessus a été écrite dans le cas d'un ordre arithmétique impliquant la mémoire. Dans le cas d'autres ordres (par exemple décalage à gauche ou à droite, ou un ordre conditionnel), l'étape II est simplifiée puisque la section emplacement de l'« interpréteur d'ordre » n'est pas utilisée. [...]

Ordres initiaux

De ce qui vient d'être dit, et par un examen du code des ordres, on voit que le mécanisme d'entrée est contrôlé par des ordres de programme. À moins qu'il y ait quelques ordres dans la mémoire par lesquels commencer, rien ne peut être pris en entrée et la machine ne peut pas démarrer. Pour cette raison, il existe une suite d'ordres – connus sous le nom d'ordres initiaux – câblés de façon permanente dans un ensemble d'unisélecteurs (interrupteurs téléphoniques rotatoires). Ces ordres peuvent être transferrés à la mémoire en appuyant sur un bouton.

Il y a une latitude considérable dans le choix des ordres initiaux mais, une fois câblés sur les unisélecteurs, il n'est pas facile de les changer.

[WR-50]

5. *Pulse* dans le texte originel.

7.3.2 Aspect logiciel

7.3.2.1 L'un des premiers langages machine

Donald KNUTH a décrit en 1970 [Knu-70] une analyse de deux des premiers jeux d'instructions conçus pour des ordinateurs à programme enregistré.

> *Le premier rapport préliminaire de von Neumann* [Von-45]*, à propos de l'EDVAC, proposait de construire un ordinateur séquentiel comprenant une mémoire principale de 3 registres de 32 bits et une mémoire secondaire de 8 192 mots de 32 bits. Les trois registres ont été originellement nommés i et j (pour l'entrée* [*input* en anglais] *de la circuiterie arithmétique) et o (pour la sortie* [*output* en anglais]*) mais, pour des raisons pratiques, nous les dénoterons par la suite, respectivement, I, J et A. La mémoire de l'EDVAC devait être divisée en 256 « banques »* [*tanks* en anglais] *de 32 mots, chaque banque fonctionnant en cycle. Le mot 0 de chaque banque devait passer devant le dispositif de lecture un bit par unité de temps à la fois, donc 32 unités de temps plus tard le mot 1 de la banque devenait disponible, ainsi de suite jusqu'au mot 31, puis à nouveau le mot 0. Ainsi le procédé d'accès aux données était en gros identique à celui disponible aujourd'hui sur les pistes des disques durs. L'unité de temps était la µsec, donc le cycle total pour une banque était de* $32 \times 32 = 1\,024$ *µsec.*
> [...]
> *Chaque mot de 32 bits était soit un nombre (premier bit du mot à 0), soit une instruction (premier bit du mot à 1). Von Neumann suggéra de représenter les nombres en binaire en les inversant, c'est-à-dire le bit de poids faible à gauche, car, de toutes façons, la notation binaire n'étant pas très familière, la convention habituelle pouvait être ignorée et surtout parce que les circuits séquentiels sont plus adaptés à calculer en commençant par les bits les moins significatifs. Le dernier bit d'un nombre (situé à la suite du dernier bit significatif) servait à représenter le signe. La représentation utilisée était le complément à deux. Ainsi le mot :*

$$b_0 b_1 b_2 b_3 \dots b_{30} b_{31}, \qquad b_0 = 0,$$

> *désignait le nombre :*

$$2^{-30} b_1 + 2^{-29} b_2 + 2^{-28} b_3 + \cdots + 2^{-1} b_{30} - b_{31},$$

> *donc les fractions sur 30 bits dans l'intervalle* $-1 \leq x < 1$ *étaient représentables. Pour l'addition, la soustraction et les opérations de conversion, le nombre pouvait être vu comme l'entier :*

$$b_1 + 2b_2 + 4b_3 + \cdots + 2^{29} b_{30} - 2^{30} b_{31},$$

> *donc les entiers dans l'intervalle* $-2^{30} \leq x < 2^{30}$ *étaient représentables. Les entiers représentés en binaire codé décimal* $(abcdefg)_{10}$ *étaient aussi utilisables, sous la forme :*

$$0000 a_1 a_2 a_3 a_4 b_1 b_2 b_3 b_4 \dots g_1 g_2 g_3 g_4,$$

> *où* $a_1 a_2 a_3 a_4$ *était le code du chiffre a, et* $b_1 b_2 b_3 b_4$ *le code du chiffre b, etc., le tout en ordre binaire inversé (donc* $0000 = 0$, $1000 = 1$, $0100 = 2$, ..., $0001 = 8$, $1001 = 9$*).*
> *Les mots représentant des instructions devaient être de la forme :*

$$1 a_0 a_1 a_2 a_3 a_4 b_0 b_1 b_2 0000000000 y_0 y_1 y_2 y_3 y_4 x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7,$$

où $a = a_0a_1a_2a_3a_4$ désignait une opération donnée, $b = b_0b_1b_2$ l'une de ces variantes, $y = y_0 + 2y_1 + 4y_2 + 8y_3 + 16y_4$ la position d'un mot dans une banque et $x = x_0 + 2x_1 + \cdots + 128x_7$ le numéro de banque. Les codes suivants correspondent aux opérations arithmétiques proposées affectant les registres I, J, et A :

- *AD* ($a = 00000$). *Effectue $A \leftarrow I + J$.*
- *SB* ($a = 00001$). *Effectue $A \leftarrow I - J$.*
- *ML* ($a = 00010$). *Effectue $A \leftarrow A + I \times J$ (avec arrondi).*
- *DV* ($a = 00011$). *Effectue $A \leftarrow I/J$ (avec arrondi).*
- *SQ* ($a = 00100$). *Effectue $A \leftarrow \sqrt{J}$ (avec arrondi).*
- *II* ($a = 00101$). *Effectue $A \leftarrow I$.*
- *JJ* ($a = 00110$). *Effectue $A \leftarrow J$.*
- *SL* ($a = 00111$). *Si $A \geq 0$, effectue $A \leftarrow I$; si $A < 0$, effectue $A \leftarrow J$.*
- *DB* ($a = 01000$). *Place dans A l'équivalent binaire du nombre décimal I.*
- *BD* ($a = 01001$). *Place dans A l'équivalent décimal du nombre I.*

(Comme nous l'avons indiqué dans l'introduction, nous avons changé la notation originelle de von Neumann : les mnémoniques utilisés pour nommer les opérations sont des symboles choisis de façon ad hoc *et dans le but unique d'améliorer la lecture. On peut noter que la multiplication, la division et la racine carrée sont supposées produire des valeurs* arrondies. *Mais les détails de ces opérations ne sont pas tous décrits dans le mémo ; par exemple, s'il y est indiqué que la division et l'extraction de racine carrée devaient changer le contenu du registre I, rien n'indique si le reste de l'opération devait y être stocké. Les opérations de conversions (décimal-binaire ou binaire-décimal) n'y sont pas été décrites non plus. Apparemment, les différents dépassements de capacité ne provoquaient rien de particulier.)*

Chacune des opérations arithmétiques ci-dessus pouvait être employée en utilisant l'une des nombreuses variantes disponibles et spécifiée par $b = b_0b_1b_2$:

- *H* ($b = 111$). *Efffectue l'opération décrite et conserve le résultat dans A.*
- *A* ($b = 100$). *Effectue l'opération et réalise les affectactions $J \leftarrow I$, $I \leftarrow A$, $A \leftarrow 0$.*
- *S* ($b = 000$). *Effectue l'opération, enregistre le résultat A à l'emplacement mémoire yx puis réalise l'affectation $A \leftarrow 0$.*
- *F* ($b = 101$). *Effectue l'opération, enregistre le résultat dans le mot mémoire situé immédiatement derrière cette instruction, réalise l'affectation $A \leftarrow 0$, puis exécute l'instruction suivante qui vient d'être modifiée.*
- *N* ($b = 110$). *Effectue l'opération, enregistre le résultat dans le mot mémoire situé immédiatement après cette instruction, réalise l'affectation $A \leftarrow 0$, puis saute l'instruction suivante.*

Ainsi, par exemple, ADS yx a pour effet de placer la valeur $I + J$ à l'adresse yx et de positionner A à 0 ; JJA a pour effet d'échanger les contenus de I et J et de réinitaliser A à 0 ; SLH a pour effet de donner à A la valeur de I ou J, selon que le signe de A est 0 ou 1. (L'adresse mémoire yx n'est utilisée que dans la variante S.)

Outre les opérations arithmétiques, la machine pouvait réaliser les choses suivantes :

- *JMP* ($a = 11000, b = 000$). *Exécute l'instruction située à l'adresse yx (puis $1 + yx$, etc.).*
- *LOD* ($a = 10000, b = 000$). *Réalise l'affectation $J \leftarrow I$, puis affecte la valeur contenue à l'adresse yx à I.*

Les autres codes $a = 01010$, 01011, 01100, 01101, 01110 et 01111 étaient réservés pour des opérations d'entrées-sorties (lesquelles n'étaient pas spécifiées) et pour arrêter la machine.

Les opérations telles que nous les avons décrites avaient une particularité très impor-

tante : seuls des nombres pouvaient être contenus dans les registres I, J et A, pas des instructions. Lorsqu'une opération LOD spécifiait une adresse mémoire contenant une instruction, seule la partie yx de cette instruction était chargée dans I ; les autres bits étaient remis à 0. Inversement, lorsqu'une affectation concernait la mémoire, par l'emploi d'une variante S, F ou N, et dans le cas où cette mémoire contenait une instruction, seuls les 13 bits les moins significatifs du nombre A étaient enregistrés dans la partie yx.

Les instructions devaient être exécutées dans l'ordre de leur emplacement en mémoire, à moins que la séquence ne soit modifiée par une instruction JMP. Si jamais le contrôle parvenait sur un nombre (et non pas sur une instruction), l'effet devait être le même qu'une instruction LOD sur ce nombre.

7.3.2.2 Le premier langage symbolique (1949)

Bien que KNUTH utilise un langage symbolique pour décrire le langage machine de VON NEUMANN, celui-ci est suffisamment satisfait par le langage machine pour ne pas avoir à se servir d'un langage symbolique.

Le langage symbolique correspondant au langage machine du premier ordinateur, l'EDSAC, est décrit par Maurice WILKES dans une conférence tenue en juin 1949 à l'université de Cambridge lors de l'inauguration de l'EDSAC. C'est un article que nous avons déjà cité puisque c'est le premier article qui décrit la réalisation effective du principe des programmes enregistrés :

Introduction

L'EDSAC (electronic delay storage automatic calculator) est une machine à calculer électronique séquentielle travaillant en base deux et utilisant des conteneurs à ultrasons pour le stockage. Le stockage principal consiste en 32 conteneurs, chacun d'à peu près 5 pieds de long et contenant 32 nombres de 17 chiffres binaires, l'un d'eux étant le chiffre de signe. Ceci donne 1 024 emplacements de stockage en tout. Il est possible d'utiliser deux emplacements de stockage adjacents ensemble de façon à obtenir un nombre de 35 chiffres binaires (incluant un chiffre de signe) ; ainsi, à tout moment, le stockage peut contenir un mélange de nombres longs *et* courts. *Des conteneurs courts, pouvant contenir un seul nombre seulement, sont utilisés comme accumulateur et comme registres multiplieurs dans l'unité arithmétique ainsi que pour le contrôle dans diverses parties de la machine.*

Un code à adresse unique est utilisé pour l'EDSAC, les ordres ayant la même longueur que les nombres courts. Le jeu complet des ordres est le suivant :

Jeu d'ordres de l'EDSAC

A n Ajouter le nombre de l'emplacement de stockage n à l'accumulateur.

S n Soustraire le nombre de l'emplacement de stockage n de l'accumulateur.

H n Transférer le nombre de l'emplacement de stockage n dans le registre multiplieur.

V n Multiplier le nombre de l'emplacement de stockage n par le nombre du registre multiplieur et l'ajouter à l'accumulateur.

N n Multiplier le nombre de l'emplacement de stockage n par le nombre du registre multiplieur et le soustraire de l'accumulateur.

T n Transférer le contenu de l'accumulateur à l'emplacement de stockage n et effacer l'accumulateur.

Table 2 Edsac Character Codes

Perforator		Teleprinter		Binary	Decimal
Letter shift	Figure shift	Letter shift	Figure shift		
P	0	P	0	00000	0
Q	1	Q	1	00001	1
W	2	W	2	00010	2
E	3	E	3	00011	3
R	4	R	4	00100	4
T	5	T	5	00101	5
Y	6	Y	6	00110	6
U	7	U	7	00111	7
I	8	I	8	01000	8
O	9	O	9	01001	9
J		J		01010	10
π		Figure Shift		01011	11
S		S	"	01100	12
Z		Z	+	01101	13
K		K	(	01110	14
Erase[1]		Letter Shift		01111	15
Blank tape[2]		(no effect)		10000	16
F		F	$	10001	17
θ		Carriage Return		10010	18
D		D	;	10011	19
ϕ		Space		10100	20
H	+	H	£	10101	21
N	-	N	,	10110	22
M		M	.	10111	23
Δ		Line Feed		11000	24
L		L	)	11001	25
X		X	/	11010	26
G		G	#	11011	27
A		A	-	11100	28
B		B	?	11101	29
C		C	:	11110	30
V		V	=	11111	31

Notes
1 Erase is represented by an asterisk ("*") in the simulator. When this character is *output*, it sets the teleprinter into letter shift.
2 Blank tape is represented by a period (".") This character has no effect on output.
3 The personal computer text environment has only a "new line" character. On the Edsac simulator, the line-feed character is interpreted as a newline character, and carriage returns are thrown away.
4 The symbols θ, ϕ, Δ or π are typed as @, !, & and #, respectively.

FIGURE 7.14 – Le code de l'EDSAC (1949)

U n *Transférer le contenu de l'accumulateur à l'emplacement de stockage n sans effacer l'accumulateur.*

C n *Collationner le nombre de l'emplacement de stockage n avec le nombre du registre multiplieur,* i.e. *ajouter un « 1 » à l'accumulateur aux positions où les deux nombres ont un « 1 » et un « 0 » aux autres positions.*

R 2^{n-2} *Déplacer le nombre dans l'accumulateur de n places à droite,* i.e. *le multiplier par* 2^{-n}.

L 2^{n-2} *Déplacer le nombre dans l'accumulateur de n places à gauche,* i.e. *le multiplier par* 2^{n}.

E n *Si le nombre dans l'accumulateur est plus grand au sens large que zéro, exécuter la fois suivante l'ordre qui se trouve à l'emplacement n ; sinon procéder séquentiellement.*

G n *Si le nombre dans l'accumulateur est plus petit au sens large que zéro, exécuter la fois suivante l'ordre qui se trouve à l'emplacement n ; sinon procéder séquentiellement.*

I n Lire la ligne suivante de trous sur le ruban et placer les 5 chiffres en résultant dans les positions de poids faible de l'emplacement de stockage *n*.

O n Imprimer le caractère en attente sur le téléscripteur et mettre en attente sur le téléscripteur le caractère représenté par les cinq chiffres de poids fort de l'emplacement de stockage *n*.

F n Placer les cinq chiffres qui représentent le prochain caractère à imprimer par le téléscripteur dans les cinq positions de poids fort de l'emplacement de stockage *n*.

X Arrondir le nombre dans l'accumulateur avec 16 chiffres binaires.

Y Arrondir le nombre dans l'accumulateur avec 34 chiffres binaires.

Z Arrêter la machine et faire sonner la clochette d'avertissement.

Nous verrons que la plupart des ordres consiste en une partie fonctionnelle qui définit l'opération et une partie numérique qui définit l'emplacement de stockage ; quelques ordres, cependant, consistent uniquement en une partie fonctionnelle.

Un ruban perforé à cinq perforations ordinaire du type de ceux utilisés en télégraphie est utilisé pour les entrées. Chaque ligne de perforations représente un nombre binaire à cinq chiffres et l'opération d'entrée de base consiste à transférer ce nombre en mémoire[6]*. De même le mécanisme de sortie est un téléscripteur et l'opération de sortie de base consiste à transférer un nombre binaire de 5 chiffres au téléscripteur et d'imprimer le caractère correspondant. Le code du téléscripteur est choisi de façon telle que les nombres binaires jusqu'à neuf soient imprimés comme chiffres décimaux correspondants et un code analogue est utilisé pour l'entrée. Ceci permet à l'opération de conversion du et vers le système décimal d'être programmée comme partie du calcul.*

Le but de l'ordre F est de permettre de vérifier l'opération du téléscripteur. À part ça, aucune fonctionnalité de vérification n'est fournie sur l'EDSAC, laissant au programmeur le soin d'incorporer dans le programme les vérifications qu'il juge nécessaire.

[WR-50]

Maurice Wilkes dit que le code du télescripteur est modifié mais ne dit rien sur cette modification. Celui-ci est publié dans le guide de l'émulateur de l'EDSAC [Cam-01] et reproduit à la figure 7.14.

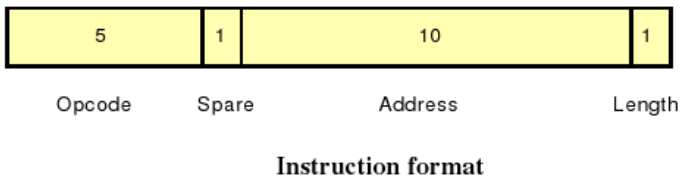


Figure 7.15 – Format d'instruction de l'EDSAC ([Cam-01], p. 14)

L'EDSAC utilise un format d'instruction à une seule adresse montré sur la figure 7.15 sur 17 bits. Le code opération (« fonction » dans le vocabulaire de l'époque) est spécifié sur 5 bits et l'adresse sur 10 bits. Un bit supplémentaire spécifie la longueur de l'opérande. La représentation binaire du code opération est la même que le code du caractère correspondant pour simplifier la traduction du programme écrit en langage symbolique. Une instruction est toujours écrite en

6. *Store* dans le texte originel.

langage symbolique à la façon 'A 56 S', qui signifie « Ajouter le nombre court de l'emplacement 56 à l'accumulateur ». Lorsqu'une adresse est zéro, elle est omise en langage symbolique.

7.4 Bibliographie

[BN-71] Bell, C. Gordon, Newell, Allen, **Computer Structures : Readings and Examples**, McGraw-Hill, 1971, XIX + 668 p. Disponible sous forme électronique :

`http://research.microsoft.com/users/gbell/Computer_Structures_ _Readings_and_Examples/contents.html`

[Bur-46] Burks, Arthur W., Goldstine, Herman H., von Neumann, John, *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument*, U. S. Army Ordnance Department, 1946. Reproduit dans Taub, A. H. (ed.), **Collected Works of John von Neumann**, Macmillan, New-York, 1963, vol. 5, pp. 34–79, dans [Swa-76], pp. 221–259 et dans [BN-71], pp. 92–119.

[Bur-80] Burks, Arthur W., *From ENIAC to the stored-program computer*, in [Met-80], pp. 311–344.

[Cam-00] Campbell-Kelly, Martin, *Past into Present : The EDSAC Simulator*, in [RH-00], pp. 297–416.

[Cam-01] Campbell-Kelly, Martin, **EdsacPC : A Tutorial Guide to the EDSAC Simulator**, July 2001, disponible à l'adresse :

`http://www.dcs.warwick.ac.uk/~edsac/`

[Eck-44] Eckert, J. Presper, **Disclosure of Magnetic Calculating Machine**, 29 January 1944. Reproduit dans [Eck-80], pp. 537–539.

[Eck-80] Eckert, J. Presper, *The ENIAC*, in [Met-80], pp. 525–539.

[Héron] Héron d'Alexandrie, **Pneumatica**, circa 300 après J.-C. Traduction française par Guillaumin, Jean-Yves et Argoud, Gilberte, **Les pneumatiques d'Héron d'Alexandrie**, Lavoisier, 1997, 200 p. Traduction anglaise par Woodcroft, Bennet, **The Pneumatics of Hero of Alexandria**, London, 1851, 111 pages, 50 illustrations, disponible en ligne :

`http://www.history.rochester.edu/steam/hero/`

[IBM-47] IBM, **IBM electric punched card machines, principles and operation. Electric Multiplier Type 601**, IBM Corp., New-York, 1947.

[Knu-70] Knuth, Donald E., *Von Neumann's First Computer Program*, **Computing Surveys 2**, 1970, pp. 247–260. Réimprimé comme chapitre 12 de *Selected Papers on Computer Science*. Traduit par Jean-Baptiste Yunès, *Le premier programme informatique de von Neumann* dans Donald E. Knuth, **Éléments pour une histoire de l'informatique**, articles choisis et traduits par Patrick Cégielski, Lecture Notes 190, CSLI Publications, Stanford, et Société Mathématique de France, 2011, xvi + 371 p.

[Lyn-47] Lyndon, Roger C., *The Zuse Computer*, **Mathematical Tables and Other Aids to Calculation**, vol. 2, octobre 1947, pp. 355–359.

[Mau-79] MAUCHLY, John W., *Amending the ENIAC Story*, **Datamation** 25 (octobre 1979), pp. 217-219.

[Met-80] METROPOLIS, N. and HOWLETT, J. and ROTA, Gian-Carlo, eds, **A History of Computing in the Twentieth Century**, Academic Press, 1980.

[Ran-82] RANDELL, Brian, **The origins of Digital Computers**, Springer, 1984.

[RH-00] ROJAS, Raúl, HASHAGEN, Ulf, **The First Computers : History and Architectures**, The MIT Press, 2000, XII + 457 p., ISBN 0-262-68137-4.

[Ste-81] STERN, Nancy, **From ENIAC to UNIVAC : An appraisal of the Eckert-Mauchly Computers**, Digital Presse, Bedford, Mass., 1981.

[Swa-76] SWARTZLANDER, Earl E., Jr, **Computer Design Development : Principal Papers**, Hayden, 1976, 310 p., ISBN 0-8104-5988-4.

[Von-45] VON NEUMANN, John, **First Draft of a Report on the EDVAC**, contract N^o W-670-ORD-492, Moore School of Electrical Engineering, University of Pennsylvania, 30 juin 1945, 101 p. Extraits dans [Ran-82], pp. 383–392. Reed. in [Ste-81], pp. 177–246 et avec des commentaires dans **Annals of the History of Computing**, vol. 15, 1993, n^o 4, pp. 27–75. Version électronique disponible à l'adresse :

`http://www.virtualtravelog.net/entries/2003-08-TheFirstDraft.pdf`

[WR-50] WILKES, Maurice V., RENWICK, W., *The EDSAC*, in **Report of a Conf. on High Speed Automatic Calculating Machines, 22-25 June 1949**, Univ. Math. Lab., Cambridge, England, Jan. 1950, pp. 9–11. Reproduit dans [Ran-82], pp. 417–421.

[Wil-71] WILKES, M. V., *Babbage as a Computer Pioneer*, **Report of Proceedings, Babbage Memorial Meeting, London, 18 Oct., 1971**, British Computer Soc., London, 1971.