Esquisse d'un ordinateur à billes pour la démonstration publique des concepts fondamentaux de l'informatique

Pierre Cubaud

Esquisse d'un ordinateur à billes pour la démonstration publique des concepts fondamentaux de l'informatique

Pierre Cubaud <cubaud@cnam.fr>
Equipe Interactivité pour Lire et Jouer, CEDRIC
Conservatoire national des arts et métiers, Paris
oct. 2021

Résumé : Un ordinateur mécanique utilisant des billes comme moyen de stockage de l'information pourrait servir de démonstrateur commode pour les espaces muséologiques consacrés à l'informatique. Nous présentons l'architecture d'une telle machine, librement inspirée du ACE d'A. Turing, puis des programmes types ainsi qu'un échéancier pour la réalisation de l'ordinateur.

1. Introduction

Lorsqu'on entre dans le musée des sciences de Londres, on est en quelque sorte accueilli par une imposante machine à vapeur en action. Au Palais de la découverte de Paris, c'est une grande machine électrostatique qui attire le public depuis des décennies. Quel pourrait être l'équivalent dans un musée consacré à l'informatique ? L'aspect immatériel du fonctionnement interne de l'ordinateur exécutant son programme est une difficulté bien connue des enseignants du domaine, dont fait partie l'auteur de ces lignes. C'est aussi, il nous semble, une difficulté pour la didactique muséale de ce domaine. On peut évidemment avoir recours à des animations sur écran, éventuellement interactives voire en réalité virtuelle. Il est possible aussi de construire des maquettes animées, qui matérialisent mieux que des écrans les objets techniques en action (mémoires, processeurs, périphériques, etc.). Il nous parait possible d'aller plus loin que ces simulations en utilisant des procédés mécaniques entièrement visibles et directement manipulables. Dans le cas de démonstrations publiques, la lenteur des procédés de calcul mécanique et leur encombrement physique sont en fait des avantages par rapport aux circuits intégrés des ordinateurs modernes. On peut aussi s'attendre à ce qu'un tel ordinateur, occupant nécessairement une dizaine de mètres cubes, présente un aspect spectaculaire comparable aux machines mentionnées plus haut.

Un cahier des charges raisonnable pour notre machine serait d'avoir une organisation générale et un mode de fonctionnement proche des systèmes électroniques intégrés, afin qu'elle puisse jouer pleinement son rôle de démonstrateur en étant mise en rapport avec les collections de l'espace muséal. Il nous semble donc important que le démonstrateur repose sur : 1) un codage binaire de l'information 2) des instructions et données stockées conjointement en mémoire 3) des instructions de rupture de séquence : décision, itération 4) des organes d'opération (arithmétique, etc.) et d'échange avec le monde extérieur. Tous les mécanismes sous-jacents devraient être entièrement visibles pour le visiteur, donc il faudra viser au minimalisme dans la réalisation. Un compromis devra cependant être trouvé pour ne pas rendre les applications de la machine trop triviales.

Il nous semble qu'un ordinateur mécanique utilisant des billes de deux types comme bits d'information pourrait satisfaire ces contraintes. Il parait simple de réaliser de cette manière des registres (série) de mots assez longs et en assez grand nombre pour permettre l'exécution de programmes non anecdotiques. Nous avons représenté fig. 1 diverses combinaisons

d’octets à partir de matériaux usuels. On peut supposer qu’un registre de 16 bits constitué de billes de 30mm occuperait une forme de tube d’environ 60 cm de longueur qui serait visible et déchiffrable à quelques mètres de distance. Le transfert par gravité des billes d'un organe à l'autre de l'ordinateur peut être rendu assez rapide (0.1s par ex.), sans pour autant être inaccessible aux sens humains. Les billes sont aussi employées dans de nombreux jeux pour tout âge (billard, flipper, etc.). Elles véhiculent donc une image d'amusement qui peut bénéficier au démonstrateur dans son ensemble.

Figure 1. Variantes de matériaux pour les bits. De haut en bas : buis et hêtre 15mm, hêtre 15 et 20mm, verre 16mm et buis 15mm, hêtre et acier 15mm, verre 16mm et acier 15mm

2. Etat de l’art

Il n'existe pas à notre connaissance d'ordinateur mécanique (ou électromécanique) utilisant des billes comme moyen de stockage de l'information, ni de publication y faisant explicitement référence. Le domaine de la micro-fluidique aborde ce sujet, mais avec des motivations et des problématiques trop éloignées des nôtres [1]. Il existe peut-être des brevets, par exemple dans l'esprit de [2]. Aucune proposition d'ordinateur mécanique ne peut évidemment ignorer l'œuvre de C. Babbage. L'architecture et la programmation de son *Analytical Engine* ont fait l'objet de nombreuses études, en particulier [3, 4]. Nous ne nous sommes intéressés qu'à trois catégories de travaux et de produits : les descriptions d'ordinateurs à objectif purement pédagogiques, les procédés de traitement de l'information basés sur l'utilisation de billes et les dispositifs purement ludiques. Même avec ces restrictions, nous ne saurions prétendre à l'exhaustivité et un état de l'art détaillé reste donc à faire.

2.1. Ordinateurs à finalité pédagogique

Les enseignants de l'informatique ont très vite ressenti le besoin d'exposer les principes de fonctionnement des ordinateurs en faisant le plus possible abstraction du détail des spécificités des réalisations commerciales (voir la machine *CODA* de [5] et le *BOULIX* de [6]).

Peu d'auteurs sont cependant allés plus loin que décrire des machines de papier[1]. D.J. Nessin décrit en revanche dans [7] l'ordinateur *STUPIDD* : une machine 4 bits pipelinée construite sur une base de composants LSI (registres, ALU) complètement opérationnelle et avec tout le détail de sa microprogrammation[2].

Pour un public plus large que les étudiants en informatique, un effort de simplification supplémentaire est nécessaire. Avant l'invention des mémoires intégrées, les machines que pouvaient acquérir ou construire les amateurs étaient sérieusement limitées par leurs capacités mémoire (quelques bits). On trouvera par exemple dans [8] la description d'un ordinateur minimal constructible, baptisé *SIMON*. C. Shanon lui-même a conçu en 1961 une machine de ce type sous le nom *Minivac 601* [9]. L'ouvrage d'Alcosser et al. [10] fournit le plan détaillé d'une machine basée sur des composants très simples (papier, aluminium, trombones etc.). L'arrivée dans les années 1975-85, de calculatrices programmables puis de micro-ordinateurs vraiment abordables a bien sûr complètement changé le panorama des machines pour l'apprentissage de l'informatique. Les manuels de ces machines sont parfois d'une bonne qualité pédagogique, comme ceux de la Ti-57 [11] et du ZX81 [12]. Malgré leur grande variété, une constante de ces produits nous semble résider dans leurs fonctions ludiques et d'interaction humain-machine souvent très développées : une idée à conserver pour notre projet (fonctions d'aléa, graphisme, son, manettes, joysticks, etc.).

Trois projets de la décennie écoulée méritent une mention particulière, car ils semblent motivés par des considérations très semblables aux nôtres. Construite en 2012 par une équipe d'enseignants et d'étudiants de l'ENS Lyon, la machine *RubENS* est une matérialisation strictement mécanique du modèle théorique de Turing [13]. La machine a été entièrement construite en Lego avec des transmissions pneumatiques. M. Reynaud a construit lui aussi une matérialisation du modèle de Turing, électromécanique celle-ci, qui fait l'objet de nombreuses présentations publiques [14]. Le site de M. Reynaud inclut de nombreux exemples de programmation. Évidemment, lorsqu'il est mis en œuvre matériellement, le modèle de Turing conduit à des programmes très lents à exécuter, ce qui limite l'ampleur des problèmes traités en pratique. Sa matérialisation est de plus loin d'être triviale, comme le prouve les descriptions des deux machines citées. Le *Megaprocessor* de J. Newman est pour sa part un ordinateur électronique d'architecture "von Neuman" conventionnelle, construit uniquement en transistors discrets [15, 16]. Avec 42370 transistors, la machine occupe tous les murs d'une belle pièce au *Centre for Computing History* de Cambridge (UK). Elle peut fonctionner à 1 Hz et chaque organe est éclairé par LED quand il est sollicité. La machine compte 8 registres et seulement 256 bits de RAM, mais elle dispose d'un jeu d'instructions très riche.

2.2. Machines à billes pour le traitement de l'information

Deux machines à billes ont été conçues explicitement pour l'enseignement du traitement automatique de l'information digitale. Il ne s'agit pas à proprement parler d'ordinateurs, car les instructions n'y sont pas stockées en mémoire et ne sont pas modifiables, mais les traitements qu'elles permettent sont suffisants pour apprendre des rudiments de programmation. La machine *Digicomp II* a été commercialisée dans les années 1960 [17]. Son

[1] Un inventaire est en cours sur Wikipedia. Il faudrait le compléter avec les ouvrages francophones mentionnés ici : https://en.wikipedia.org/wiki/Category:Educational_abstract_machines

[2] L'auteur a eu la bonne fortune d'avoir M. Nesin comme professeur en 1985-6 et possède encore cette machine.

manuel, très riche, explique comment calculer par itération les termes d'une série infinie (p. 51) et donne un exemple de génération aléatoire et de simulation (pp. 55-57). La machine *Turing Tumble* a été conçue par A. et P. Boswell dans les années 2010 et est actuellement commercialisée [18]. La machine se destine spécialement à l'enseignement. Elle est accompagnée elle aussi d'un livret très bien conçu.
Le dispositif clé, commun à ces machines, est le *flip-flop*. Il est en effet très simple de réaliser une bascule d'états binaire contrôlée par la chute de billes (fig. 1). A chaque passage de bille, la bascule change de position et dirige la bille suivante dans une nouvelle direction. Le flot d'entrée est ainsi divisé en deux parties égales.

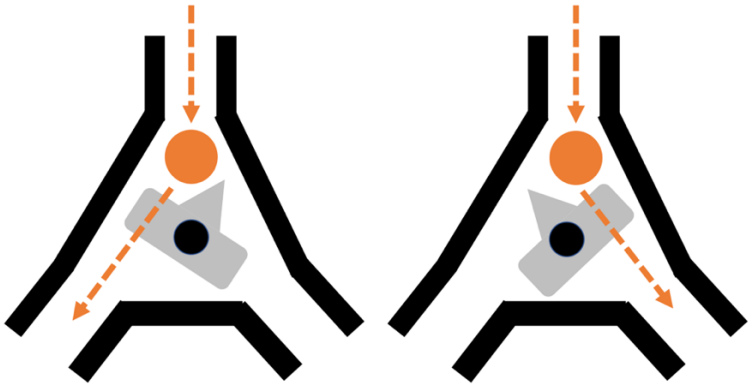


Figure 1 - Flip-flop à bille. A gauche : premier passage de bille. A droite : passage de la bille suivante.

L'autre point commun à ces machines est un dispositif répéteur : la sortie d'un réservoir de billes, en haut de la machine, est asservie à un levier situé en bas, qui est actionné par l'arrivée des billes en fin de parcours. Ainsi, il est possible d'enchainer des traitements sur l'étendue du plan de travail. Le *Turing Tumble* compte deux types de billes (bleues et rouges) et deux répéteurs. Les deux machines diffèrent aussi par le type d'opérateurs actionnés par les billes.

Le *Marble Adder* de M. Wandel réalise la fonction d'addition binaire avec une modification très intéressante du flip-flop. On trouvera sur le site de M. Wandel [19] des vidéos de démonstration, ainsi que des plans d'exécutions très détaillés. Seules les billes de valeur 1 interviennent dans l'opération et, au résultat, les bits 0 sont représentés par une absence de bille. Le mécanisme de M. Wandel a été repris dans de nombreuses réalisations documentées sur Internet mais sans amélioration notable, à notre avis. L'addition binaire avec des billes a aussi été réalisée par M. Donahoe et J. DeCew en utilisant le jeu de construction *K'nex* [20]. Dans cet imposant additionneur 4 bits, les deux types de billes/bits coexistent. Une synthèse des deux machines est sans doute possible pour notre projet. Un opérateur spécifique pour l'incrémentation serait également utile pour le compteur de programme et pour réaliser l'opération de complément à deux de la soustraction.

2.3. Dispositifs ludiques à billes

Il existe de très nombreux jeux de construction permettant de bâtir des circuits de billes. La plupart sont en matière plastique, les billes étant guidées par des rails ou des tubes. Il existe des variantes en bois, mais aux circuits plus simples. Dans presque tous les produits, on retrouve une variante de la bascule flip-flop, des circuits ralentisseurs (entonnoir) ou accélérateurs (looping). Les produits les plus sophistiqués sont sans doute ceux de la société

FischerTechnik. Ils sont en effet compatibles avec la vaste gamme de jeux de construction pour l'enseignement technique mis au point par cette société, incluant des composants électromécaniques, pneumatiques ou numériques.
Le développement conjoint du partage d'information via l'internet et des procédés de fabrication numérique a permis également à toute une communauté d'amateurs de développer de très ingénieux jouets à billes. Les vidéos abondent sur ce thème et un catalogue raisonné reste à constituer : voir par exemple les œuvres du pseudonyme Denha [21]. Nous terminons ce rapide inventaire avec la magnifique réalisation de M. Olin, du groupe suédois *Wintergatan*. Il s'agit d'un instrument de musique animé par billes dans l'esprit des boîtes à musique. Pilotée par la « partition » d'un grand cylindre à picots reconfigurable, la chute de chaque bille entraine la production d'une note par percussion de divers éléments. M. Olin fédère un vaste groupe d'amateurs afin de réaliser une nouvelle version de sa machine, la *Marble Machine X* [22].

3. Architecture de l'ordinateur à billes

L'architecture présentée ici est le résultat de plusieurs itérations de l'auteur, mais il ne s'agit encore que d'une esquisse qui devra être confrontée aux observations et idées de spécialistes. Nous avons effectué une revisite des ordinateurs en mode série de première génération basés sur des mémoires à mercure (EDVAC, etc.), pour conclure que l'architecture de la "vraie" machine de Turing, le *ACE* (*Automated Computer Engine*), pouvait être transposée à grands traits pour notre machine. Une spécificité du *ACE* est que son format d'instruction n'a pas de code d'opération [23, 24]. Il inclut deux références : celle de l'unité source d'information et celle de l'unité destination. Les réservoirs (*tanks*) de mémoire sont mis sur le même plan que les unités de traitement et échangent leurs données via un bus unique, sans usage d'accumulateurs. Le compteur de programme (PC) et le registre d'instruction (IR) sont les seuls registres en plus des réservoirs. Il nous parait intéressant de garder une telle organisation, qui simplifie l'unité de contrôle. Le *ACE* ne disposait que de 11 instructions (dont une reconfigurable par microprogrammation), suffisantes toutefois pour que Turing propose une large gamme de problèmes solubles par la machine [25].
L'architecture retenue est décrite sur la figure 2. On y retrouve les principaux organes du *ACE* : PC et IR, registres de mémoires (MEM) et organes opérateurs (OPER). Nous ne faisons à ce stade aucune hypothèse sur le nombre d'organes d'opération et de mémoires (notés respectivement m et n sur le schéma).[3] L'exécution des programmes est effectuée par l'automate CONTROL. Les transferts entre registres mémoires, de mémoire vers opérateur et d'opérateur vers mémoire sont traités de manière quasi identique par CONTROL. Celui-ci commande aussi les vannes d'entrée et de sortie de PC et de IR. L'instruction courante est stockée dans IR et décodée en ses deux parties : source et destination. Ces deux mots sont transmis à deux démultiplexeurs jouant le rôle de sélecteur de source (OUT_SEL) et de destination (IN_SEL). Le microprogramme de CONTROL se déroule classiquement en trois étapes : placement dans IR de l'instruction issue de la case mémoire référencée par PC puis exécution de l'instruction et enfin mise à jour de PC (par incrémentation ou rupture de séquence). Si l'instruction exécutée a pour destination un organe opérateur, celui-ci le « découvre » par l'ouverture de sa vanne d'entrée. Il signale alors son démarrage puis, en fin d'opération, sa terminaison à CONTROL, qui attend pendant cette période.

[3] Le nombre total d'organes m+n doit bien sûr être cohérent avec la taille de l'espace adressable dans les instructions, elles-mêmes dépendant du nombre de bits b accordé aux mots mémoire : m+n ≤ 2^(b/2)

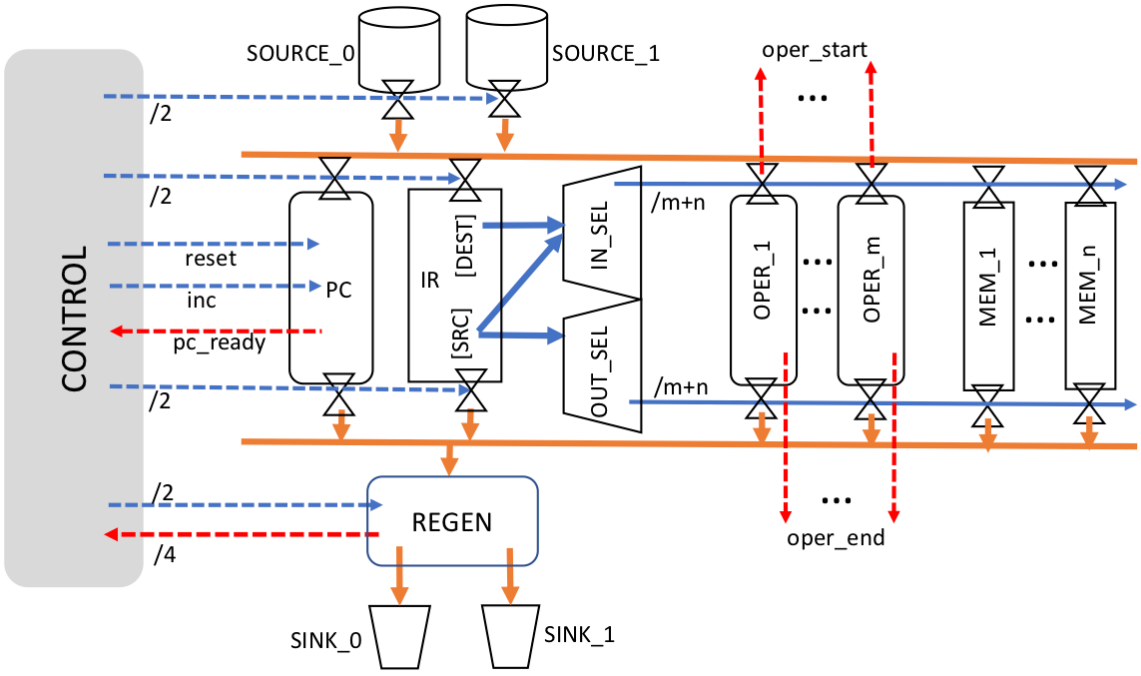


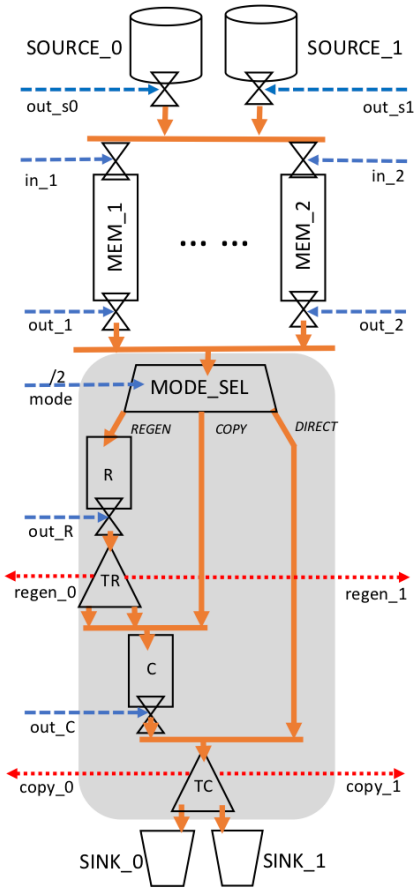
Figure 2. Organisation générale. Les bus de données/billes sont indiqués en orange, les commandes principales de l'unité de contrôle sont indiquées en bleu, les signalisations en retour sont indiquées en rouge.

Notre machine a aussi des différences importantes avec le *ACE*. Les mémoires à billes doivent être régénérées comme les *tanks* à mercure, mais seulement lors d'une demande de lecture. C'est le rôle du module REGEN dont le fonctionnement sera détaillé ci-après. Il faut de plus disposer d'un réservoir de billes pour gérer la modification des valeurs des bits (modules SOURCE_0 et SOURCE_1). Pour éviter de consommer des billes infiniment, il parait prudent de chercher à les recycler. Les modules SINK_0 et SINK_1 jouent le rôle de réceptacles, qu'un humain devra régulièrement vider dans les sources. Nous n'avons pas non plus retenu l'organisation en bus circulaire unique du *ACE* afin d'utiliser uniquement la gravité pendant le transfert d'information d'une unité à l'autre. Les billes circulent des modules SOURCE en haut, aux SINK placés au point bas de la machine. Le transfert d'une unité source vers une unité de destination provoque la régénération de l'information dans l'unité source puis sa copie dans la destination.

La partie grisée sur la fig.3 détaille l'organisation interne de REGEN, le module de régénération. On y trouve deux tampons à ouverture commandées (R et C), un sélecteur 3 voies (MODE_SEL) et deux « discriminateurs » (TR et TC). Ces derniers modules ont pour fonction de reconnaitre la valeur des bits/billes qu'ils reçoivent et de les aiguiller dans des directions différentes, tout en signalant le passage de chaque bille à l'unité de contrôle. Le microprogramme pour un transfert mémoire-mémoire est résumé sur la figure. Le transfert depuis un opérateur se déroule de la même manière. Le transfert vers un opérateur peut ne pas nécessiter de régénération, auquel cas MODE_SEL est positionné sur sa sortie « copy ». La phase de recherche de l'instruction courante nécessite deux transferts : de PC vers RI puis de MEM à nouveau vers RI. Ces deux opérations peuvent elles aussi être effectuées selon le microcode de la fig. 3.

Les trois sélecteurs de la machine (IN_SEL, OUT_SEL et MODE_SEL) sont des démultiplexeurs qui, selon la consigne d'un mot de n bits, aiguillent le trajet d'une bille parmi 2^n chemins

possibles. Ces « aiguillages » peuvent être matérialisés par des bascules commandées par un axe externe en translation, dont la direction dépend de la valeur du bit de commande (fig. 4). Pour MODE_SEL, ce sont des billes de données qui circulent dans le sélecteur. Pour IN_SEL et OUT_SEL, il s'agit d'une bille de commande dont la chute en bas du dispositif va déclencher l'ouverture de la vanne du dispositif sélectionné en source ou en destination. La bille de commande doit ensuite être recyclée.




```
//// régénération de MEM_1
-> in_1
-> MODE_SEL : REGEN
répéter
        -> out_1
jusqu'à MEM_1 vide
répéter
    -> out_R
    <- regen_0 ; -> out_s0 | <- regen_1 ; -> out_s1
jusqu'à R vide

//// vider MEM_2
-> MODE_SEL : DIRECT
répéter
        -> out_2
        <- copy_0 | <- copy_1
jusqu'à MEM_2 vide

//// copier MEM_1 dans MEM_2
-> in_2
répéter
    -> out_C
    <- copy_0 ; -> out_s0 | <- copy_1 ; -> out_s1
jusqu'à C vide
```

Figure 3. Détail du module de régénération et microprogramme pour un transfert mémoire/mémoire. Le symbole -> indique un signal émis par CONTROL (en tirets bleu sur le schéma). Le symbole <- correspond à un signal reçu par CONTROL (tirets rouge). Les symboles ; et | indiquent une séquence d'événements et des événements en parallèle.

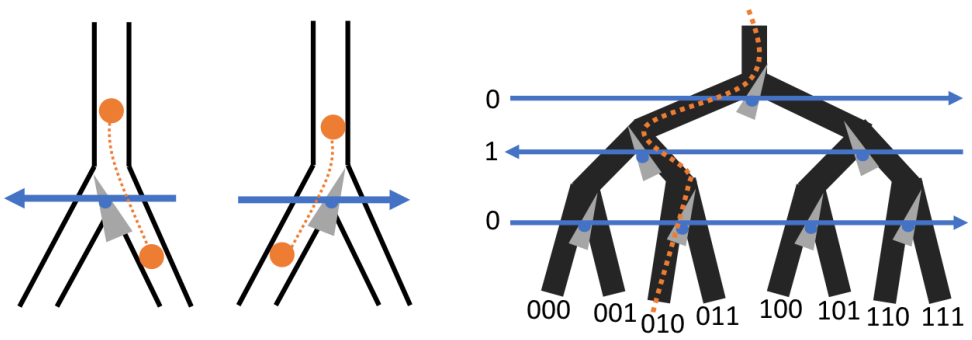


Figure 4. Principe d'un sélecteur à billes, agissant comme démultiplexeur 1 bit (à gauche) et 3 bits (à droite).

Il reste à préciser le fonctionnement des modules discriminateurs TR et TC. Ces modules doivent séparer les bits entrants selon leur valeur, afin que soit envoyée une bille de copie de même type par les modules SRC en amont. La discrimination de valeur des bits peut reposer sur de nombreuses caractéristiques physiques : le diamètre des billes, leur masse volumique, leur conductibilité, etc. Des tests différents pourraient de plus être combinés pour une plus grande fiabilité. A titre d'exemple, nous avons réalisé un prototype rudimentaire utilisant l'attraction magnétique (fig. 5). Les billes entrantes roulent sur une planche inclinée de haut en bas (pitch = 10°) et de gauche à droite (roll = 2°). Les billes d'acier (diam. 15mm) sont attirées par un groupe d'aimants qui dévie leur trajectoire vers la gauche. Les billes de verres (diam. 16mm) sont en revanche déviées vers la sortie de droite, qui est dans le sens de la pente. Le dispositif s'avère fiable, avec une répétabilité constatée de 320 passages consécutifs sans erreur[4]. Il est cependant très sensible aux trajectoires initiales des billes, à l'emplacement exact des aimants et aux angles d'inclinaison. Il est aussi assez lent, avec un débit d'une bille pour 2 s env.

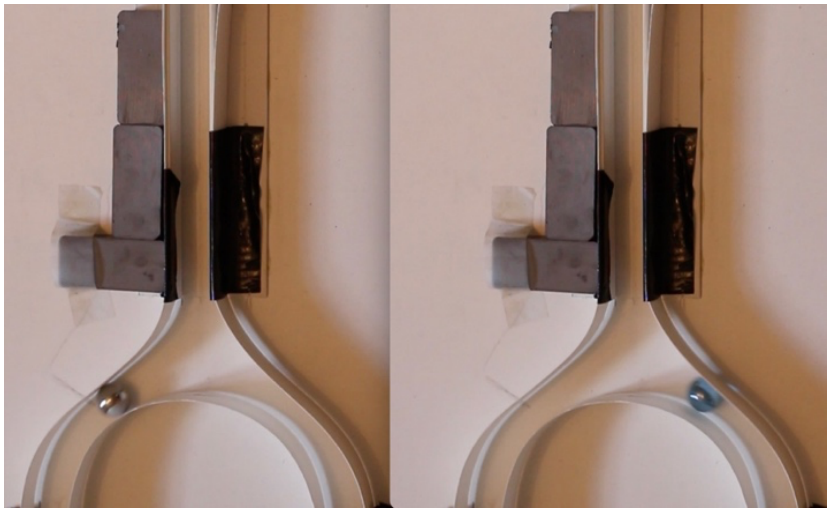


Figure 5. Preuve de concept d'un discriminateur à aimant de billes d'acier (à gauche) et de verre (à droite). Une vidéo est disponible sur https://youtu.be/HrTJB-h-z1k

Nous n'avons rien spécifié à ce stade concernant les technologies à employer pour la transmission des commandes et des signalisations, ni sur la nature de l'énergie motrice employée pour l'activation des vannes. Il faudra trancher entre des procédés strictement mécaniques (utilisant l'énergie cinétique des billes, des bascules, des ressorts, etc.), des procédés électromécaniques (solénoïdes, micro-rupteurs, etc.) ou pneumatiques comme pour la machine *RubENS*. S'agissant de l'unité de contrôle elle-même, il nous parait possible de recourir, comme l'imaginait C. Babbage, à un ou plusieurs cylindres à picots (semblables à ceux des boites à musique) pour produire la séquence des signaux qui matérialisent le microprogramme. Une réalisation très convaincante d'une telle unité de contrôle a été produite en Meccano par T. Robinson dans les années 2000 [23]. La machine musicale de M. Olin déjà citée est également une bonne piste d'investigations [22]. La rotation des cylindres sera elle-même cadencée par une horloge mécanique à pendule et pourra être interrompue par les signaux reçus des opérateurs, ou par une intervention humaine.

[4] Limite de la patience de l'auteur-testeur. On est ici bien sûr très en deçà des exigences de fiabilité d'une machine opérationnelle.

4. Programmation et performance de l'ordinateur à billes

4.1. Jeu d'instruction

La définition de la partie opérative d'un ordinateur dépend toujours en pratique de la classe d'utilisation visée : traitement de signal, manipulation intensive de textes, machine à pile, etc. Notre démonstrateur doit au contraire mettre l'accent sur l'aspect arbitraire de la sémantique des opérations qu'effectue un ordinateur. A notre avis, une instruction produisant un son de cloche ou un jet de peinture sur un tableau apporte autant pour la compréhension de la programmation et du fonctionnement d'ensemble de l'ordinateur qu'une addition. Le temps limité que peuvent consacrer les visiteurs au démonstrateur et la lenteur prévisible de notre machine nous incite par ailleurs à restreindre au maximum le nombre d'opérateurs et de types de données. On se limitera, comme souvent dans les machines de papier citées plus haut, à un unique format pour les données : les entiers signés. Une dizaine d'opérateurs s'avèrent indispensables afin que des notions essentielles de programmation soient exposées : le traitement conditionnel et la répétition. Des notions plus avancées comme le codage des nombres réels, le mécanisme de sous-programme ou l'adressage indirect pourraient aussi être prises en compte ou, au contraire, reportées sur des démonstrateurs annexes.

Il nous semble aussi important que notre démonstrateur éclaire certaines situations d'usage quotidien, en particulier ce qui concerne le dialogue humain-machine et la connexion des ordinateurs en réseau. On pourrait imaginer que l'ordinateur à billes soit accessible via un site web, à partir duquel les visiteurs peuvent soumettre un mot binaire et, inversement, visualiser un mot extrait de la mémoire. Un tel procédé peut bien sûr générer une file d'attente de requêtes, qu'il faudrait scénariser dans l'esprit, par exemple, de l'installation artistique *Surexposition* de S. Bianchini [27].

Le tableau 1 récapitule les opérations que nous avons sélectionné pour l'ordinateur à billes. On rappelle que les instructions sont de la forme unique <destination, source>. Nous avons choisi des identifiants semblables à ceux d'assembleurs que nous avons utilisés dans le passé.

4.2. Exemples de codes

Le jeu d'instruction de la table 1 a été choisi en partie pour permettre d'aborder un classique de l'initiation à la programmation : le problème de Syracuse (ou conjecture de Collatz, ou encore suite 3A+1). On trouvera dans [28] un exemple de progression pédagogique sur ce sujet, permettant d'aborder les notions de décision et d'itération. La suite de Syracuse est définie par récurrence de la manière suivante :

$$\forall\, A_0 \in \mathbb{N}^*, A_{n+1} = \begin{cases} A_n/2 \; si \; A_n \; est \; pair \\ 3A_n + 1 \; sinon \end{cases}$$

On conjecture que la suite se termine toujours par le cycle 4,2,1. Par exemple, pour la valeur initiale 127, on trouve la séquence assez riche suivante : 127, 382, 191, 574, 287, 862, 431, 1294, 647, 1942, 971, 2914, 1457, 4372, 2186, 1093, 3280, 1640, 820, 410, 205, 616, 308, 154, 77, 232, 116, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

Le problème peut être abordé par un programme qui demande à l'utilisatrice la valeur initiale de la suite puis itère jusqu'à obtenir 1. On donne fig. 6 le code assembleur correspondant pour l'ordinateur à billes, ainsi que l'encodage binaire. On a pris ici des mots de 16 bits, avec comme convention que les billes de verre codent un bit 0 et les billes d'acier un bit 1. L'adresse choisie pour les opérateurs est donnée table 2. Le programme requiert 17 instructions et 3 cases mémoires pour les variables. Il n'est pas sûr que tous les visiteurs soient attirés par un

programme de ce type, à connotation théorique, mais celui-ci peut servir de "benchmark" pendant la mise au point de l'ordinateur.

Catégorie	DEST	SRC	Opération
Calcul	ACC	src	acc := [src]
	dest	ACC	[dest] := acc
	ADD	src	acc := acc + [src]
	OPP	src	opp := -[src]
	dest	OPP	[dest] := opp
	SHR	src	shr := [src]/2
	dest	SHR	[dest] := shr
	RND	src	rnd := randomisation des bits 1 contenus dans [src]
	dest	RND	[dest] := rnd
Séquence	END	—	fin d'exécution (pas de paramètre source)
	GTO	n	signal à CONTROL pour PC := n
	TST	src	tst := [src]
	GTZ	n	si tst = 0, signal à CONTROL pour PC := n
	GTN	n	si tst < 0, signal à CONTROL pour PC := n
	GTE	n	si tst est pair, signal à CONTROL pour PC := n
Dialogue	HUM	src	affiche [src] sur écran en base dix
	dest	HUM	attente de saisie clavier et transfert à [dest]
	EXT	src	envoi de [src] à la machine externe
	dest	EXT	attente de données ext. et transfert à [dest]
	BELL	src	produit un son fonction des bits de [src]

Table 1. Jeu d'instruction de l'ordinateur à billes

Opérateur	Adresse
ACC	254
ADD	253
BEL	243
END	255
GTE	245
GTN	246
GTO	249
GTZ	247
HUM	244
OPP	252
RND	250
SHR	251
TST	248

Table 2. Exemple de plan d'adressage pour les opérateurs

Figure 6. Code et binaire du programme "syracuse". Les mots binaires ont leur poids fort à gauche (acier=1, verre=0). Le symbole @ signale la valeur initiale de PC.
Le symbole # signale une donnée.

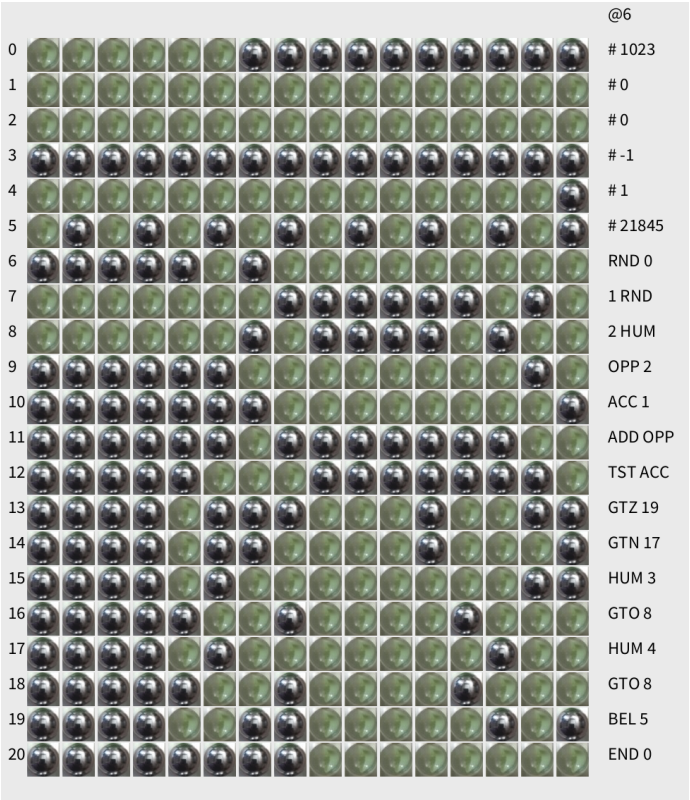


Figure 7. Code et binaire du programme "nombre mystérieux"

Pour mieux impliquer les visiteurs, on doit sans doute imaginer des programmes plus interactifs et (un peu) plus ludiques. Le "jeu du nombre mystérieux" est adapté de [11]. La machine tire au sort un nombre entier élément de [0, 1023]. L'utilisatrice doit deviner ce nombre et proposer une valeur à la machine. Celle-ci répond en retour la valeur -1 si le nombre proposé est trop petit, et +1 s'il est trop grand. Le programme s'arrête quand le nombre correct est trouvé, auquel cas une cloche sonne. On supposera donc l'existence d'un opérateur BEL pour la production sonore et d'un opérateur RND capable de remplacer, par tirage au sort, les bits 1 contenus dans un mot source[5]. Le code assembleur et le binaire correspondant sont donnés fig. 7.
De nombreux autres programmes semblent à la portée de la machine ainsi définie. On peut aborder la question du traitement de données en tables, en employant la technique bien oubliée de la réécriture d'instructions en cours d'exécution. On peut aussi par ce biais sensibiliser les visiteurs au fonctionnement des codes viraux, capable de détruire l'information contenue dans le plan mémoire, tout en assurant leur propre propagation.
En donnant au plan mémoire une taille assez généreuse (200 cases), il est possible de stocker en permanence une dizaine de programmes de démonstration du type que nous venons de décrire. Cela simplifiera sans doute l'utilisation de la machine au quotidien en limitant l'emploi d'un chargeur de programmes. Ce pourrait être aussi un bon point d'entrée pour des explications concernant le système d'exploitation, son rôle et son fonctionnement de base. Sur un plan plus fondamental, la question de la fiabilité des calculs et du stockage de l'information peut facilement être abordée avec cette machine dont chaque bit est directement visible et modifiable par une action humaine directe. Il est possible également d'évoquer l'indécidabilité de certains problèmes, en prenant l'exemple de la prévision du stock exact de billes nécessaires pour l'exécution d'un programme.

4.3. Performances attendues

Les quelques lignes de code que nous venons de présenter peuvent servir d'indicateur pour le dimensionnement et les performances souhaitables de notre ordinateur. En supposant que les discriminateurs aient un débit d'un bit/s et que la mémoire soit organisée en mots de 16 bits, un transfert mémoire-mémoire durerait environ une minute, dont 3*16 = 48s pour la régénération de la source et sa copie dans la destination - le restant correspondant à l'activation des sélecteurs. Les opérations arithmétiques et l'évaluation des conditions de branchement seront forcément plus lentes. Les transferts entre opérateurs et le branchement direct (GTO), sans régénération, seront en revanche plus rapides. L'exécution d'un cycle du programme "syracuse" pour calculer un nouveau terme de la suite correspond à une douzaine d'instructions, donc 15 min. environ. On peut supposer qu'un public motivé serait retenu sur cette durée - mais guère plus. Pour le programme interactif "nombre mystérieux", il y a 7 instructions entre chaque saisie par l'utilisateur. Si celui-ci adopte une stratégie par dichotomie, il lui faudra au pire 8 essais donc 8*7 = 56 minutes au total, ce qui est trop long.
Travailler avec des mots plus courts (12 bits ?) réduirait évidement en proportion la durée de toutes les opérations de la machine. Mais évaluer la durée de microinstruction à une seconde, comme nous l'avons fait, est aussi assez pessimiste. Une division par 10 de ce temps est sans doute difficile à atteindre sans le recours à la chute libre pour les transferts des mots/billes, qui imposerait une mécanique de contrôle sophistiquée. Une division par 3 ou 4 nous semble en revanche un objectif réaliste et suffisant pour la qualité de l'expérience utilisateur.

[5] On pense ici au Quincux de Galton. Le discriminateur à aimant de la fig. 5 est une version biaisée du Quincux.

5. Une feuille de route pour le projet

Pour aller au-delà de l'esquisse présentée ici, nous devrons mener un ensemble d'investigations sur les principes de la mémorisation de l'information par billes. La technologie de la mémoire principale des ordinateurs a en effet toujours été le facteur dominant d'où découle le reste de l'architecture [29].

1) Etude de la cinétique des billes en mouvement, étude des dispositifs employés dans l'industrie pour le convoyage et le stockage de billes ou objets similaires, étude de l'offre commerciale selon matériaux et dimensions
2) Etude de la discrimination binaire. Le discriminateur verre/acier par aimant présenté plus haut doit être perfectionné, mais il faut aussi investiguer le poids, la matière, le diamètre ou toute autre caractéristique physique permettant d'atteindre un haut niveau de fiabilité (lui-même encore à définir).
3) Etude de la régénération unaire. Choix des rails ou tubes de stockage, mise au point des portes commandées pour les entrées/sorties. Recherche d'un débit convenable.
4) Réalisation d'un régénérateur binaire complet par couplage des deux dispositifs précédents.
5) Mise en place d'une fonction de transfert mémoire/mémoire complète. Ajout d'une commande en boucle du dispositif, pour mener une étude des temps d'exécution et de la fiabilité. On souhaite obtenir un temps de transfert entre 15 et 20s, idéalement avec des mots de 16 bits.

Nous réaliserons ensuite un prototype (v0) de la machine avec une unité de contrôle simulée par ordinateur externe (type RaspberryPi, Arduino, etc.). Celui-ci simulera aussi au début les actions des opérateurs, de PC et de RI.

6) Mise en place d'une mémoire 32 cases. Exécution de programmes du type de "syracuse" avec simulation des opérateurs sur l'ordinateur externe.
7) Réalisation des opérateurs les plus simples à concevoir (END, BELL, RND, SHR), test par programmes adaptés, par exemple sur le thème de la musique générative.
8) Réalisation et test des opérateurs d'addition et de soustraction (ACC, ADD, OPP)
9) Réalisation et test des opérateurs de branchement (TST, GTO, GTZ, GTN, GTE).
10) Exécution du programme "syracuse" avec tous ses opérateurs matériels
11) Mise en place du PC et de la fonction de transfert mémoire vers le RI, choix des capteurs pour le décodage d'instruction en <source> et <destination>. Si la conception du PC et de son incrémenteur s'avère trop complexe, on peut revoir la syntaxe des instructions en y ajoutant l'adresse de l'instruction suivante et celle de saut. On peut supposer suffisant d'affecter 5 bits à chacun des champs, avec donc des mots mémoires de taille 20 bits.
12) Mise en place des sélecteurs d'adresse (IN_SEL et OUT_SEL) sur 5 bits

A ce stade, l'ordinateur externe ne simulera plus que l'unité de contrôle. La version suivante (v1) de la machine devra disposer d'une unité de contrôle autonome. Il faudra donc faire le choix de la technologie définitive de transmission des commandes : mécanique, pneumatique ou électromécanique.

13) Réalisation des automates de l'unité de contrôle pour le cycle d'exécution des instructions.
14) Réalisation de l'horloge et des dispositifs d'interruption du séquencement de CONTROL.

15) Réalisation du plan mémoire définitif (de 64 à 256 cases ?) et des sélecteurs d'adresse adaptés (8 bits ?)
16) Mise en place des opérateurs HUM et E/S externe via internet.
17) Finalisation des programmes et des scénarios de démonstration.

6. Conclusion

Un ordinateur mécanique aux dimensions "spectaculaires" pourrait servir de démonstrateur au grand public des principes de base de l'informatique. Nous avons proposé pour cela une architecture reposant sur une mémoire binaire utilisant deux types de billes. Contrairement à d'autres machines existantes, les billes sont utilisées ici comme stockage de données et non pas seulement comme signal de transmission. Il parait ainsi possible de réaliser une mémoire centrale assez grande, autorisant des programmes non triviaux. En contrepartie, la solution proposée nécessite un recyclage des billes à chaque lecture en mémoire, couteux en temps. Nous avons présenté une esquisse du fonctionnement possible de la machine, ainsi que de sa programmation.

Pour pouvoir aller plus loin, il nous semble indispensable de travailler en équipe pluridisciplinaire réunissant des mécaniciens, des designers, des informaticiens et, bien sûr, des muséologues et des scénographes. Le projet pourrait sans doute être mené dans le cadre de projets d'étudiants des domaines que nous venons de mentionner et/ou d'amateurs organisés avec l'appui d'un "fablab". Le prototype de la machine pourrait être mis au point directement sous les yeux du public du musée hébergeur, sous la forme d'un atelier ouvert comme à l'Exploratorium de San Francisco. L'auteur étudiera avec grand intérêt tout commentaire et toute proposition de collaboration pour l'avancement de ce projet.

7. Références bibliographiques

[1] T.C. Draper, C. Fullertyon, N. Phillips, B.P.J. de Lacy Costello, A. Adamatzky. Mechanical Sequential Counting with Liquid Marbles. *Proc. int. conf. Unconventional computation and natural computation (UCNC'18)*. Springer, 2018.

[2] A.J. Gehring et al. *Pure fluid computer*. US patent 3,190,554 (1965).

[3] A.G. Bromley. Charles Babbage's Analytical Engine, 1838. IEEE Annals of the History of Computing. July-Sept 1982, pp. 196-217.

[4] R. Rojas. The computer programs of Charles Babbage. *IEEE Annals of the History of Computing*. Jan-March 2021, pp. 6-18.

[5] P. Naslin. *Principes des calculatrices numériques automatiques*. 3ème ed. Dunod, 1965.

[6] J.P. Meinadier. *Structure et fonctionnement des ordinateurs*. Larousse, 1988 (numérisation en ligne sur le site cnum.cnam.fr)

[7] D.J. Nesin. *Processor organization and microprogramming. A project case study*. SRA, 1985.

[8] E.C. Berkeley. *Cerveaux géants, machines qui pensent*. Dunod, 1957. (trad. A. Moles)

[9] Scientific Development Corp. *Getting acquainted with Minivac 601*. Waterton, Mass. 1961. (en ligne avec la description d'une réplique sur https://www.instructables.com/Minivac-601-Replica-Version-09/)

[10] E. Alcosser, J.P. Phillips, A. M. Wolk. *How to build a working digital computer*. Hayden book company, 1967 (2nd printing 1968, 180 p.)

[11] TI Programmable 57. *Introduction à la programmation*. Texas Instruments, 1977.

[12] S. Vickers. *ZX 81 BASIC. Cours de programmation*. Sinclair research lim., 1980.

[13] http://rubens.ens-lyon.fr/fr

[14] http://machinedeturing.com
[15] S. Cass. The Megaprocessor, the wondrous insanity of a CPU the size of a room. *IEEE Spectrum*, oct. 2016, pp 19-20.
[16] http://www.megaprocessor.com/
[17] DIGI-COMP II. *Mechanical binary digital computer. Instruction manual*. Education Science Research, sd (années 1960)
[18] https://www.turingtumble.com
[19] https://woodgears.ca/marbleadd/
[20] http://knexcomputer.blogspot.com
[21] https://www.youtube.com/user/denha
[22] Chaine vidéo en ligne *Wintergatan Wednesdays* : https://www.youtube.com/c/Wintergatan. Voir en particulier https://youtu.be/IvUU8joBb1Q (>188M vues en sept. 2021). Les plans de la machine et sa musique sont par ailleurs diffusés sur http://wintergatan.net
[23] B.E. Carpenter, R.W. Doran. The other Turing machine. *The computer journal* 20(3), jan. 1977, pp. 269-279.
[24] B.E. Carpenter. Turing and ACE - Lessons from a 1946 computer design. En ligne sur https://cds.cern.ch/record/263304/files/p230.pdf
[25] M. Campbell-Kelly. Programming the Pilot ACE: early programming activity at the National Physical Laboratory. *IEEE Annals of the history of computing* 3(2), april 1981, pp. 133-162.
[26] http://www.meccano.us/analytical_engine/index.html
[27] https://reflectiveinteraction.ensadlab.fr/surexposition/
[28] A. Engel. *Mathématiques élémentaires d'un point de vue algorithmique*. CEDIC, 1979. (Adaptation de D. Reisz), pp. 22-29.
[29] M. Williams. *History of computing technology*. IEEE Press and Wiley, 2nd edition, 1995, pp. 301-3.

Les URLS citées ont été vérifiées fin sept. 2021.