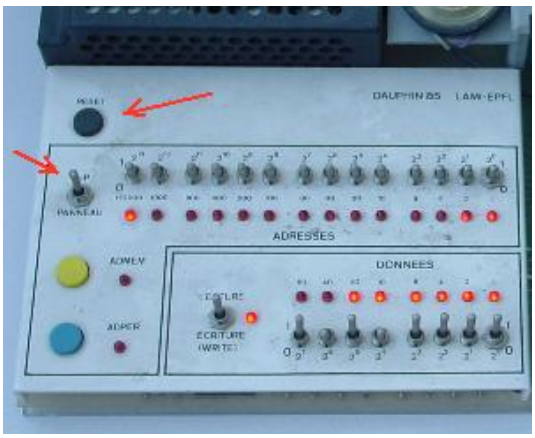
Le Dauphin 1977

Le Dauphin a été développé en 1977 pour les jeunes des clubs d'électronique.
Avec le nouveau microprocesseur Signetics 2650 qui ne coûtait que 80.- (200.- pour le 8080 ou 6800), le kit Dauphin pouvait se vendre 270.-. Quelques 150 kits ont été vendus dans le cadre du Club d'Electronique.
La carte KIM-1 (1976) de Commodore aux USA visait le même objectif d'introduction aux microprocesseurs. Elle devait être distribuée en Suisse, mais coûtait plus de 600.-
JD s'est probablement inspiré de son clavier et de son affichage hexadécimal.
Expliquer les modules d'un microprocesseur sur un bus était un avantage certain, pédagogique et commercial comme cela a réussi avec le Dauphin Industrie de Stoppani.
Le Dauphin 77 est expliqué ici de façon complémentaire à ce que vous pouvez lire sur le site Bolo: smaky.ch-dauphin

Dauphin complété par une carte écran et une cartes série/cassette.(1977)

Contrairement au KIM-1 et ses concurrents, orientés logiciels, le Dauphin voulait expliquer en détail le schéma d'un système microprocesseur et l'exécution des instructions. Le Signetics 2650 était moins performant que l'Intel 8080, le Motorola 6800 et le MOS Tech 6502; mais Il avait l'avantage de son prix et d'un répertoire d'instruction plus simple. Son point faible était la gestion des interruptions et il n'a pas convenu à l'industrie. La carte 2650 du Dauphin a été remplaçée par une carte 8080, puis Z80 et d'autres, C'était l'avantage extraordinaire du Dauphin, que JM Roullier a su exploiter avec le Dauphin de Stoppani.

Le Dauphin n'était pas un "bi-processeur" comme cela a été dit. Le panneau de test affichait et donnait accès à tous les signaux du bus par ses commutateurs. Un poussoir "reset" redémarrait le processeur comme sur tous les systèmes. Dans le mode "panneau", le commutateur sur le signal **Read-Write** interagit avec la mémoire et le clavier-affichage comme le processeur,au moment de l'action des poussoirs **SelMem** et **SelPer**. L'affichage binaire regroupait les Leds par groupe de 3. L'octal avait le grand avantage d'éviter l'apprentissage de l'hexadécimal, qui n'apportait rien à la compréhension du processeur.	

Le Dauphin a été à peine amélioré en 1986 par le LAMI. La carte processeur de ce Dauphin85 était un Motorola 68'008 et permettait de se familiariser avec les instructions performantes de la famille 68'000. L'assemblage et le téléchargement des programmes se faisait depuis le Smaky196, mais le chargement manuel de petits programmes était encore apprécié par des enseignants qui avaient récupéré des Dauphins 85 en 2020.

Comprendre le processeur est facile si on peut observer son fonctionnement en pas-à-pas. Les simulateurs n'ont su faire cela sur une station personnelle que beaucoup d'années plus tard. Chaque processeur est différent dans le détail, mais l'interface bus unifie les signaux qui pilotent les transferts. On a vu plus haut Read/Write, SelMem et SelPer.
Le signal **HoldRequest** demande au processeur de libérer le bus, comme si on enlevait la carte. Le processeur termine alors l'instruction en cours et active **HoldAcknowledge**
Le signal **NotReady** demande au processeur d'attendre que les données arrivent, ce qui peut être long s'il y a un observateur humain.

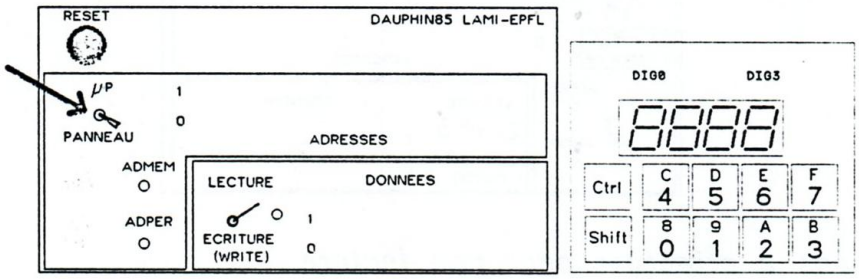
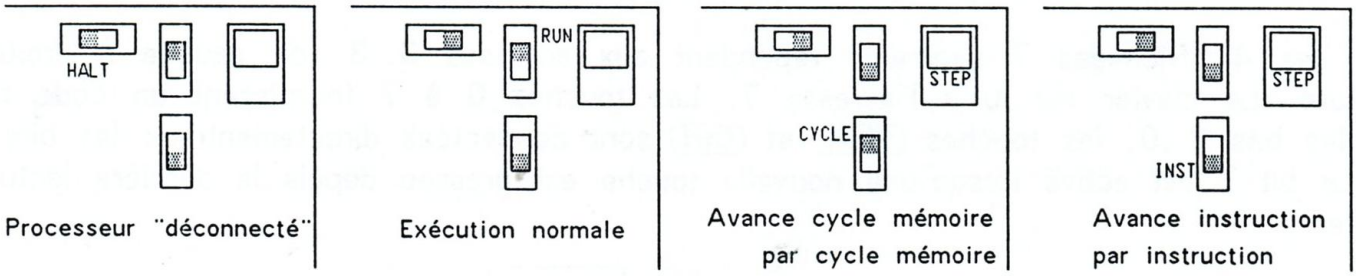
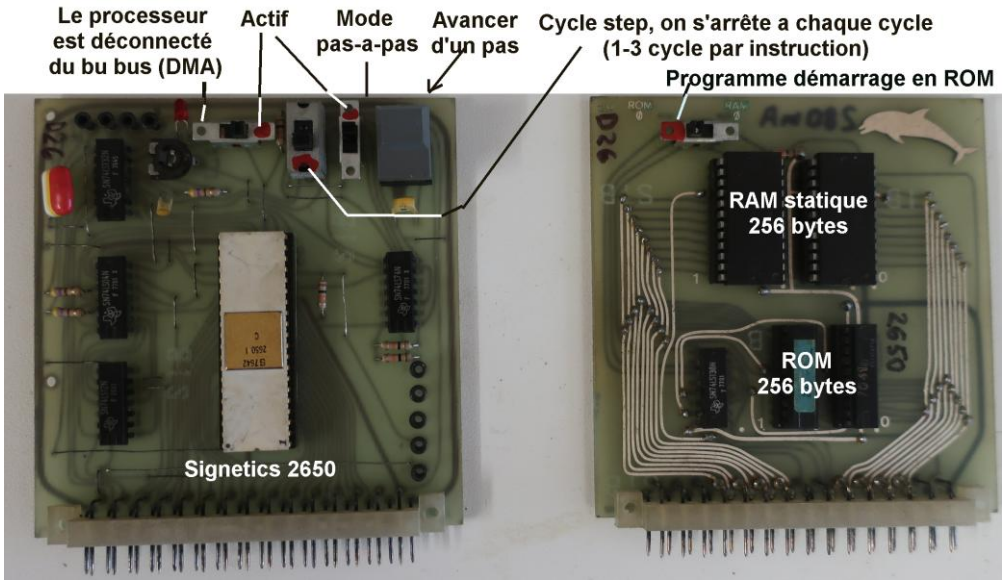


Fig. 1 Panneau et clavier du Dauphin

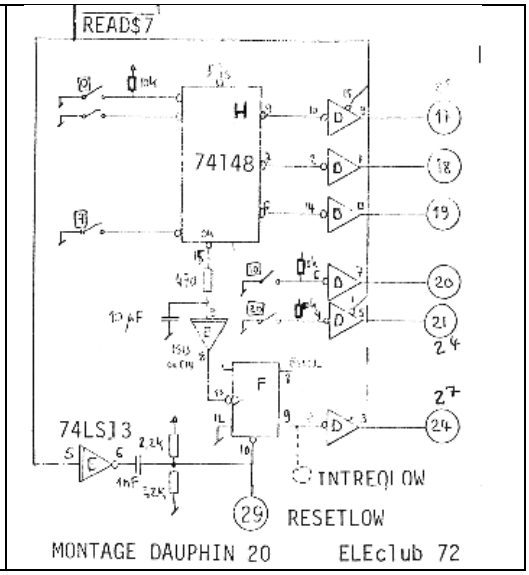


Avec ces signaux et un peu de logique, trois interrupteurs et un poussoir, permettent les modes de fonctionnements de la figure ci-dessus.
Si le panneau de test n'est pas maître, on peut lire sur ses leds les adresses, données et signaux de contrôle, à la vitesse du pocesseur ou dans un mode en pas à pas. La ROM contient le programme moniteur. Un interrupteur permet de désactiver cette ROM et on peut alors exécuter à la main à partir de l'adresse zero des instructions et tester par exemple ce qu'il faut pour lire le clavier.



Interface Clavier

Le clavieroctal se comporte comme une mémoire de 1 mot de 8 bits, en lecture et en écriture. Avec ses 10 touches, il faudrait 2 mots mémoire de 8 bits et un logiciel assez compliqué; mais l'interface serait simple.
Un circuit encodeur 74148 lit les 8 touches et rend un code 3 bits, plus l'indication qu'une touche est pressée.
Il y a 2 bits supplémentaires pour les des deux touches fonction. Ce qui n'est pas évident, c'est comment savoir qu'une nouvelle touche est pressée. Il ne faut pas compter les rebonds de contacts, ce qui se fait actuellement avec un logiciel "temps réel". La solution des années 70 est à la fois analogique et logique:
Sur la sortie "touche pressée" du 74148, un retard intègre le signal; il faut avoir pressé quelques dixième de seconde pour que le processeur le voie. Il agit et revient dans la

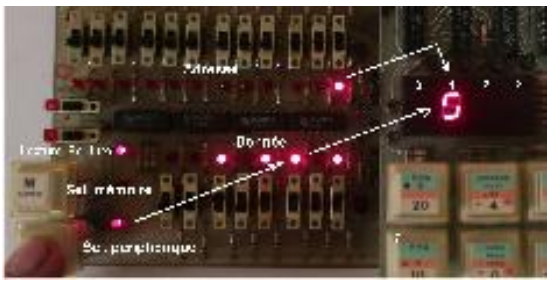


boucle qui surveille si la touches est pressée. Elle l'est toujours, puisqu'il ne s'est passé que quelques millisecondes! Si on ralenti les demandes du processeur, on rate des touches.
La solution est une bascule **FULL** activée par le 74148 et désactivée par la lecture du processeur. Quand le processeur revient de son traitement, la bascule n'a plus la mémoirede la touche précédente et il attend de nouveau. Il passe la majorité de son temps à attendre une touche, ou, comme on va le voir, il surveille le clavier dans son cycle répétitif d'affichage,

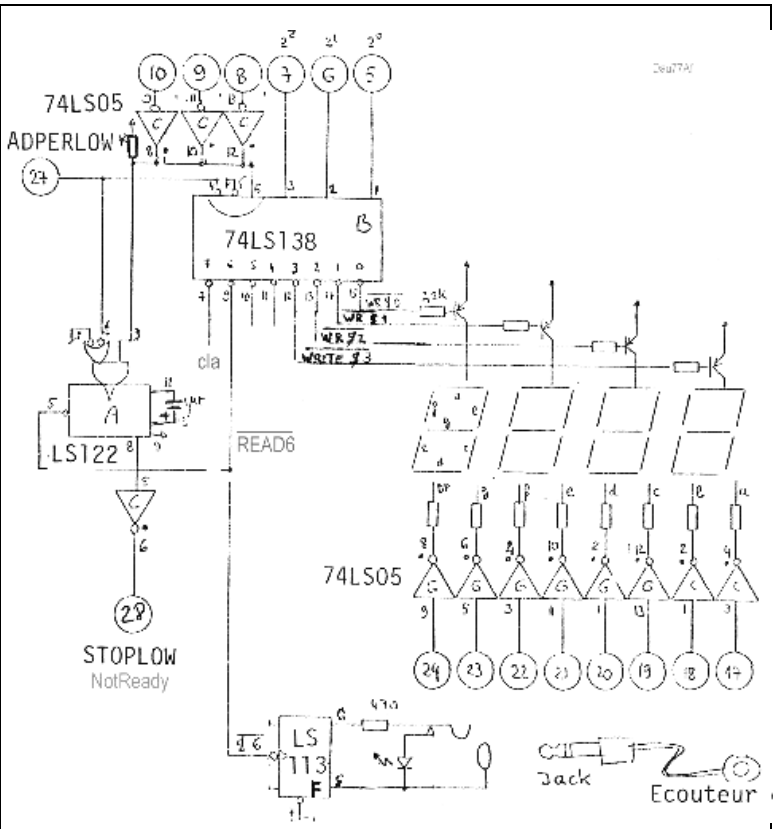
Affichage 7 segments

L'astuce de l'affichage 7 segment de Hewlett Packard, nouveauté impressionnant à l'époque, est que la luminosité est suffisant pour pouvoir multiplexer. Mais multiplexer signifie un lot d'instructions, avec une seule de temps en temps pour éclairer un affichage é la fois.
La solution du Dauphin est d'utiliser le signal **NotReady** pour prolonger, avec un circuit monostable 74LS122, la durée de la sélection. L'instruction d'écriture durait des millisecondes. Les 30 instructions pour passer au digit suivant étaient négligeables. Mais en pas-à-pas, il fallait péniblement avancer dans le programme pour voir les allumages successifs.

Le reste du schéma est "traditionnel":
Un décodeur 74LS138 sélectionne les adresse périphériques 8 bits de 70 à 77 en octal. 70 à 73 pour sélectionner les affichages, les segments sont câblés en parallèle sur le bus de donnée. Le clavier est à l'adresse 77.



Faire un peu de musique était essentiel. Le plus simple était une bascule 74113, que l'écriture de n'importe quoi à l'adresse 76 fait basculer. Il faut donc envoyer des impulsions à fréquence double. A noter que le retard du monostable est inhibé.



Analyse d'un mode d'adressage

Pour la compréhension du processeur, les instructions arithmétiques demandaient déjà à réfléchir, à cause des nombres négatifs et des dépassements de capacité.
Les modes d'adressage étaient un concept nouveau sauf pour les bibliothécaires et archivistes.
Prenons l'exemple de l'adressage relatif; le processeur Intel 8008 n'avait pas ce mode d'adressage.

Le données dont a besoin un programme sont souvent proches. Coder "25 instructions plus loin est une instruction plus courte que "à l'adresse16 bit octal 052414".
Cela ajoute des instructions et de la complexité dans le processeur. Sans entrer dans tous les détails, la figure suivante montre que le processeur doit exécuter 3 cycles, trois impulsions d'horloges. L'instruction utilise 2 octets, son code et le déplacement relatif, entre -127 et +127.
Dans un premier cycle, le processeur lit le code de l'instruction suivante et le transmet au décodeur d'instructions. Le mode "relatif" est activé.
Le cycle suivant lit et mémorise le 2^{e} byte de l'instruction.
Le 3^{e} cycle additionne l'adresse du compteur d'adresse avec le déplacement, sélectionne la donnée et la transmet au processeur, qui peut en même temps se préparer à lire l'instruction suivante.
On devine tout l'art, pour le concepteur du circuit, de gérer les aiguillages, tenir compte des délais et arriver à une fréquence d'horloge qui bat la concurrence.

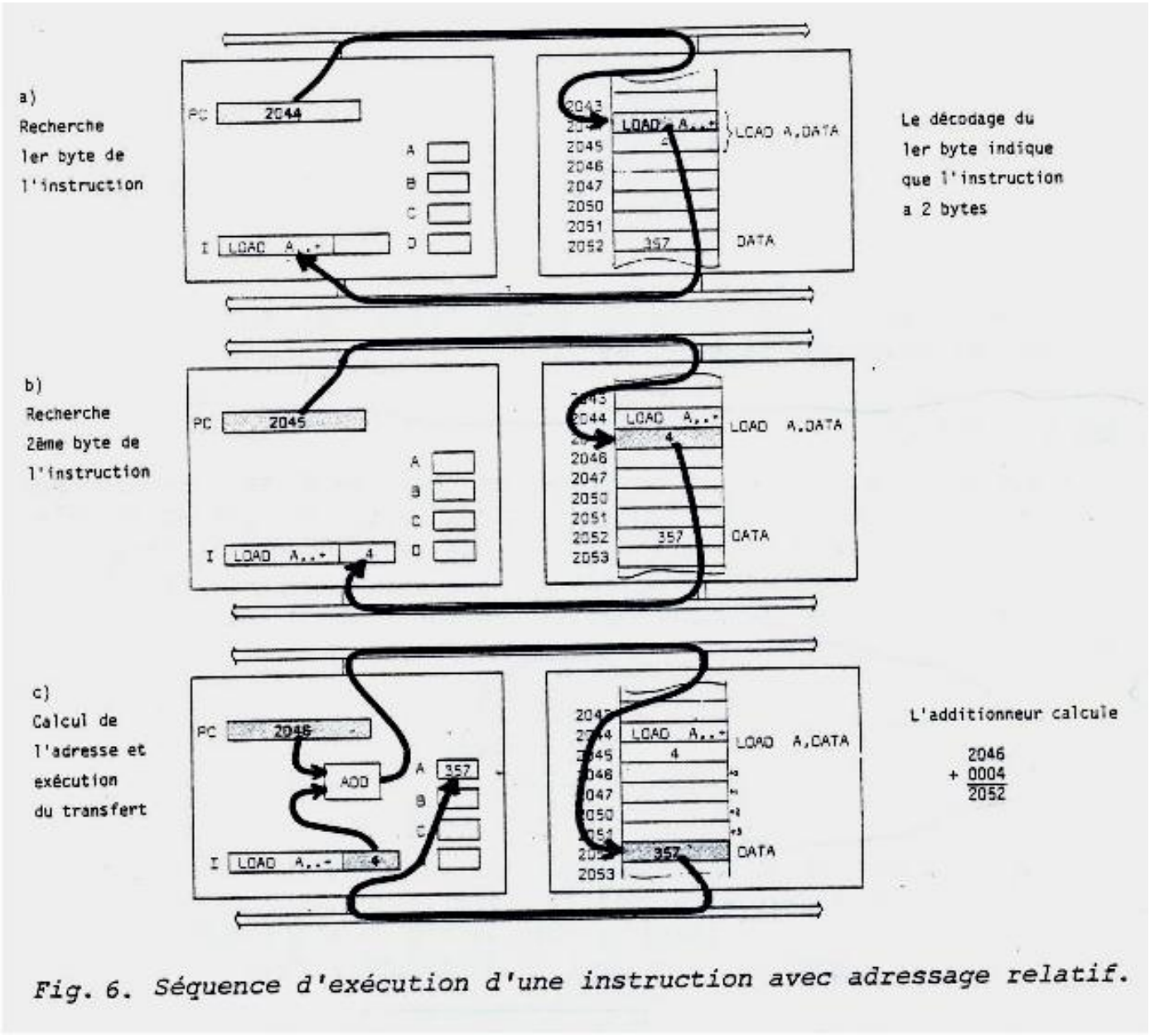


Fig. 6. Séquence d'exécution d'une instruction avec adressage relatif.

jdn 24.01.2026