Dauphin Web
Exemples de séquences pédagogiques

(N. Kunz 2024)

Table des matières

Remarques :

Pour une expérience optimale – utiliser une résolution horizontale d'au moins 1200px

Lien vers le simulateur : https://kunznicolas.github.io/Web_Dauphin/

Lien vers le repository : https://github.com/KunzNicolas/Web_Dauphin

1. Utilisation du bus

Représentation graphique et usage de la mémoire

Pour cette activité, les élèves doivent connaître la représentation des nombres binaire et hexadécimale ainsi que la notion de pixel d'une image matricielle.

Une autre activité pour aborder justement la représentation des nombres avec le simulateur est disponible en annexe (cf. chapitre Représentation des nombres). Ces activités permettent une première prise en main de l'interface du simulateur et illustrent les liens entre les différents systèmes de numération.

Les objectifs visés sont, dans un premier temps, de comprendre la représentation d'image matricielle et son lien avec l'espace mémoire en utilisant les affichages comme entrée. Puis, les processus de lecture et d'écriture en mémoire à l'aide du bus.

Le cœur de l'activité se fait au niveau des affichages (Digital et Bitmap), et de l'espace mémoire qui leur est dédié (cases grisées, cf. Figure 1). Le bus sera utilisé pour faire le lien entre ces deux composants. Il est bien sûr possible d'utiliser un des deux autres affichages.

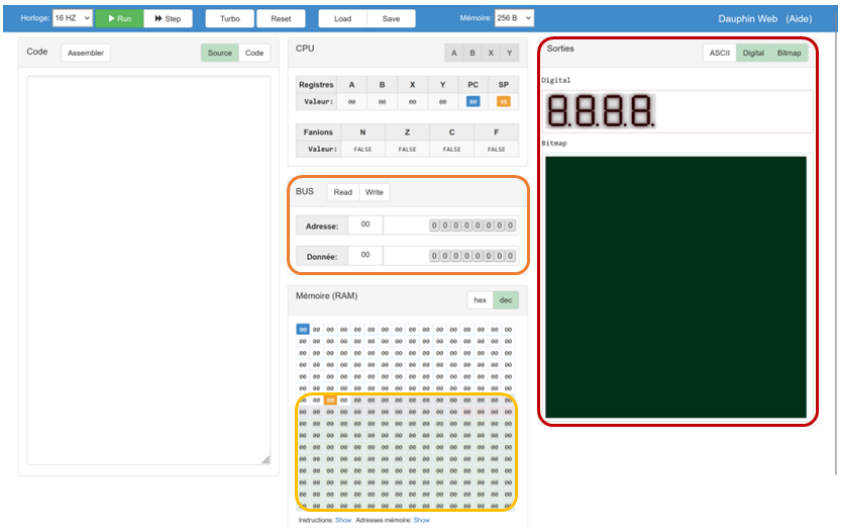


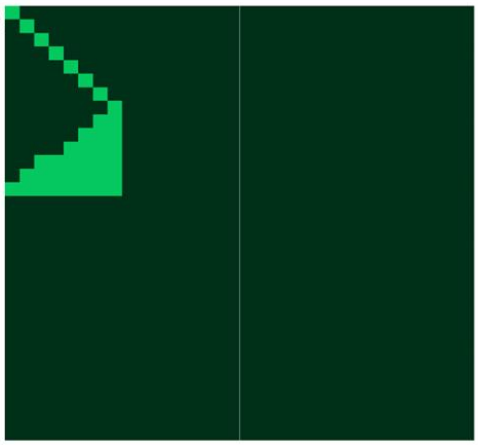
Figure 1 : Les affichages se trouvent à droite du simulateur (encadré rouge). Il est possible les cacher avec les boutons au sommet du panneau. Les cases mémoires de l'affichage sont grisées dans le panneau central mémoire (encadré jaune). Le bus en orange permet d'interagir entre ces deux éléments.

Activité 1 :

But : Comprendre le fonctionnement d'un affichage et sa représentation en mémoire. Mise en évidence du lien entre un bit mémoire et un pixel.

- Choisir soit l'affichage Digital soit Bitmap.
- Laisser les élèves découvrir l'interface en allumant et éteignant des pixels.
- Demander aux élèves de reproduire l'image suivante en allumant les pixels en cliquant dessus.

a)



b)

128	00	00	00	64	00	00	00	32	00	00	00	16	00	00	00
08	00	00	00	04	00	00	00	02	00	00	00	01	00	00	00
03	00	00	00	07	00	00	00	15	00	00	00	63	00	00	00
127	00	00	00	255	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Figure 2 : a) Exemple d'image à reproduire dans l'affichage Bitmap. b) Espace mémoire correspondante à l'image représentée à gauche.

- Faire le lien avec l'espace mémoire et l'affichage Bitmap.
- Montrer comment faire apparaître l'adresse des cases mémoires et des pixels de l'affichage en laissant la souris immobile sur la case/pixel concerné.

Activité 2 :

But : Utiliser le bus pour écrire dans la mémoire dont le changement d'état fera apparaître une image ou du texte dans un des affichages.

- Donner à l'élève une liste d'adresses mémoire et les valeurs correspondantes à y mémoriser.
- Utiliser le bus pour charger un à un chaque octet de données en mémoire grâce au bouton « Write »)

Activité 3 :

But : Transmettre à un camarade les instructions mémoires à effectuer à l'aide du bus pour reproduire une image

- Chaque élève dessine une image sur l'écran Bitmap.
- Il note sur une feuille les valeurs enregistrées en mémoire correspondant à son image.
- Il échange sa feuille avec un camarade et essaie de reproduire l'image.

2. Premières instructions

Pour cette activité, les élèves doivent avoir fait les activités précédentes sur l'Utilisation du bus. De plus, ils doivent avoir déjà quelques notions de programmation et sur la représentation de caractère (tableau ASCII).

Les objectifs visés sont de comprendre les étapes de traitement des instructions d'un processeur. Premièrement, la phase de codage des instructions en mémoire, puis leur exécution. Les élèves pourront découvrir leurs premières instructions d'assembleur.

L'activité commence par analyser les codes d'opérations de quelques instructions de base (MOVE, ADD, etc.) puis utiliser le bus pour enregistrer ces instructions dans la mémoire. Finalement, l'activité se termine par l'exécution du programme et l'affichage du résultat.

Activité 1 :

But : Utiliser le bus pour enregistrer des instructions en mémoire puis les exécuter pour afficher du texte dans l'affichage ASCII.

- Présenter aux élèves le système de codage des instructions assembleur.
- Introduire l'instruction MOVE #VAL, ADDR (codage : [DC] [vv] [ma] [aa]).
- Présenter l'affichage ASCII et les adresses mémoires rattachées
- Donner un exemple pour afficher la lettre « A » dans la première cellule de l'affichage :

Adresse	Valeur	Commentaire
H'00	H'DC	Charge le code d'opération MOVE dans la première cellule mémoire
H'01	H'41	Charge le premier opérande de l'opération MOVE qui est le code ASCII de la lettre A dans la cellule suivante
H'02	H'01	Charge la première partie de l'adresse 12bits de l'affichage ASCII (H'164)
H'03	H'64	Charge la deuxième partie de l'adresse

- Exécuter le programme (cf. Figure 3).

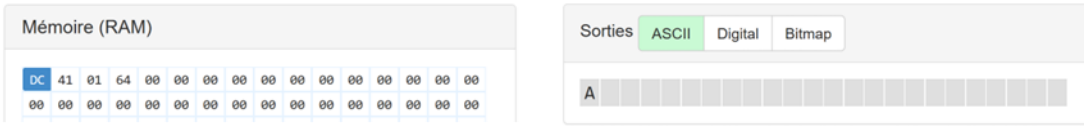


Figure 3 : Résultat de la première partie de l'activité

- Demander aux élèves d'écrire leur prénom en suivant l'exemple.
- Faire remarquer aux élèves que les codes ASCII sont ensuite copiés dans les cases mémoire de l'affichage ASCII (cf. Figure 4).

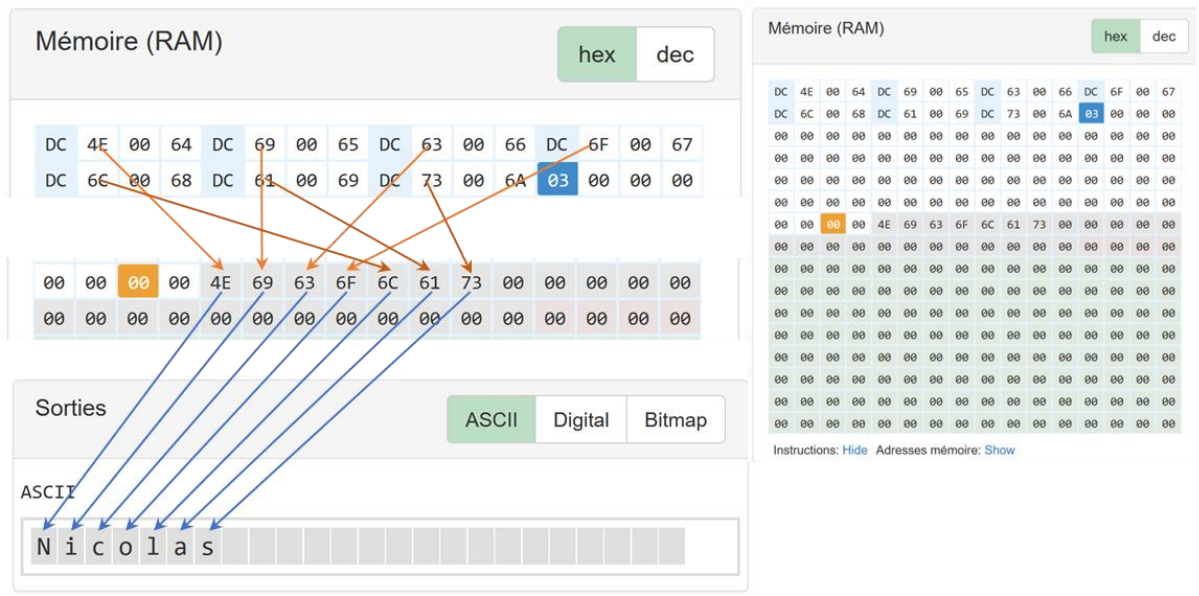


Figure 4 : À gauche, les instructions MOVE encodées dans les premières cellules mémoires reliées aux cellules de l'affichage ASCII (flèches orange). Chaque case grisée est liée à une cellule de l'affichage ASCII (flèches bleues). À droite, une vue d'ensemble de l'espace mémoire à la fin de l'exercice pour le prénom « Nicolas ».

Activité 2:

But : Créer un programme assembleur pour reproduire l'activité précédente.

- Présenter la syntaxe de l'instruction « MOVE ».
- Il existe plusieurs variantes d'abstraction pour écrire ce programme (cf. Tableau *1*).
    - En reprenant les valeurs et les adresses de l'activité précédente.
    - En utilisant directement les caractères entre guillemets au lieu de leur code ASCII.
    - Utiliser la constante _ASCII pour accéder l'adresse de la première cellule de l'affichage.

Tableau 1 *: Exemple de solution. Variante A : Bas niveau d'abstraction. Variante B : Haut niveau d'abstraction*

Variante A	**Variante B**
`MOVE #H'4E, H'64`	`MOVE #"N", _ASCII`
`MOVE #H'69, H'65`	`MOVE #"i", _ASCII+1`
`MOVE #H'63, H'66`	`MOVE #"c", _ASCII+2`
`MOVE #H'6F, H'67`	`MOVE #"o", _ASCII+3`
`MOVE #H'6C, H'68`	`MOVE #"l", _ASCII+4`
`MOVE #H'61, H'69`	`MOVE #"a", _ASCII+5`
`MOVE #H'63, H'6A`	`MOVE #"s", _ASCII+6`
`HALT`	`HALT`

- Introduire l'instruction «HALT» qui arrête le simulateur.

3. Premiers programmes

Cette activité nécessite des connaissances en informatique plus avancées et serait plus propice en fin de 1ère année de l'école de maturité ou éventuellement en 2ème année. Elle fait appel aux notions de boucle, de condition et de routine.

Les objectifs visés sont de programmer en langage machine et de faire le parallèle avec des langages de plus haut niveau sur les notions mentionnées ci-dessus. De plus, elle permet d'introduire les registres et leur rôle de variable ou de compteur par le processeur. Elle met aussi en évidence le grand nombre d'opérations élémentaires nécessaires pour réaliser une tâche décrite par quelques lignes de codes.

Activité 1:

But : créer un programme qui affiche et calcule la suite de Fibonacci dans le registre X.

- Donner la définition de la suite de Fibonacci : u[0] = 0, u[1] = 1, u[i+1]=u[i]+u[i-1]
- Pour calculer l'élément suivant de la liste, le programme a besoin de 3 variables et donc 3 registres :
  - A ← u[i-1]
  - B ← u[i]
  - X ← u[i+1]
- Donc la relation de la suite devient : X = A + B
- Ensuite, il ne faut pas oublier de mettre à jour la valeur des registres A et B :
  - A ← B
  - B ← X
- Le dernier élément nécessaire et la création d'une boucle grâce à la fonction « JUMP ».
- Plusieurs améliorations sont possibles sur la base de ce programme :
  - Utiliser un saut conditionnel pour s'arrêter avant un dépassement de capacité :
    - Le fanion « carry » passe à 1 lors d'un dépassement de capacité.
    - L'instruction « JUMP,CC » n'effectue le saut que lorsque le carry est à 0.
  - Utiliser un afficheur pour visualiser les éléments de la suite.
  - Mémoriser la suite dans la pile.

Activité 2:

But : Créer un programme qui affiche un point sur l'affichage Bitmap qui se déplace horizontalement et rebondit lorsqu'il atteint un bord.

- Les registres X et Y correspondent aux coordonnées respectivement horizontales et verticales, du point sur l'affichage Bitmap.
- Le registre B mémorise le sens de déplacement, +1 vers la droite, -1 vers la gauche.
- Les instructions de sauts jouent deux rôles distincts dans ce programme :
  - Ligne 17 : JUMP LOOP ; boucle.
  - Ligne 6 : JUMP,LO RIGHT ; condition pour que le pixel continue d'avancer vers la droite.
  - Ligne 10 : JUMP,HI LEFT ; condition pour que le pixel continue d'avancer vers la gauche.
- Les deux instructions conditionnelles sont précédées d'une comparaison sur la valeur contenue dans le registre X qui détermine si le pixel a atteint l'un des deux bords.
- Dans l'exemple donné en annexe, l'appel de la routine « _NotPixel » peut facilement être remplacé par une instruction « MOVE » :
  - Ligne 14 : MOVE #0, X
  - Ligne 16 : MOVE #1, X

- Plusieurs améliorations sont possibles sur la base de ce programme :
    - Faire que le point se déplace sur la ligne du milieu de l'affichage.
    - À chaque rebond, le point doit descendre d'une ligne.

4. Représentation des nombres

Cette activité ne nécessite aucun prérequis particulier et peut être utilisée comme introduction aux représentations binaires et hexadécimales. L'usage du simulateur est détourné pour apporter un côté ludique, ou du moins interactif, à ce thème très calculatoire.

Les objectifs d'apprentissages visés pour les élèves sont de s'approprier de manière interactive les opérations de changements de base du binaire au décimal et du binaire à l'hexadécimal.

Seul le panneau bus d'adresses et de données est utilisé lors de cette activité (cf. Figure 5).

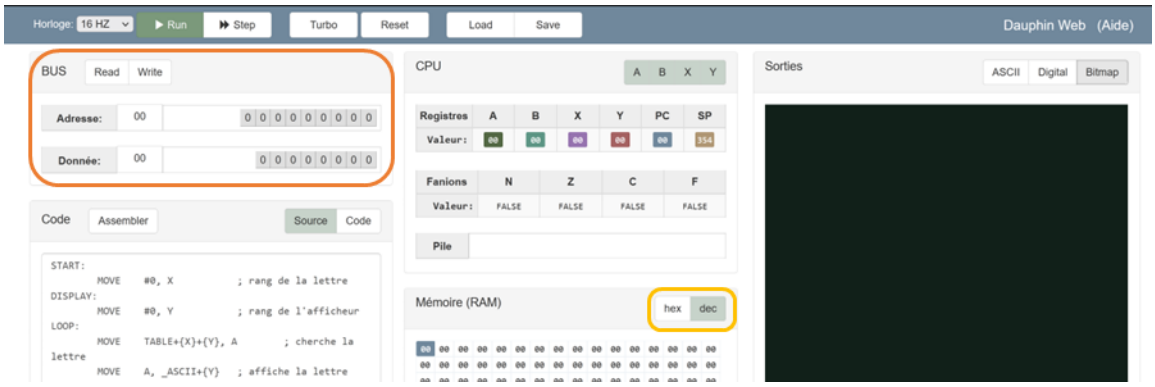


Figure 5 : Le panneau bus se trouve au sommet droit du simulateur (encadré orange). Dans l'encadré jaune, les boutons «hex» et «dec» permettent de changer de la représentation décimale à la représentation hexadécimale.

Déroulement

Introduire le simulateur aux élèves et décrire les éléments utiles à l'activité (panneau bus ainsi que le lien pour changer l'affichage en décimale ou hexadécimal, cf. Figure 5).

Activité 1 :

But : Observer les propriétés de chacune des bases et la notion de notation positionnelle.

- Demander aux élèves de remplir le tableau suivant en s'aidant du bus de données

Décimal	Binaire	Hexadécimal
0		
1		
3		
…		
16		

Activité 2 :

But : Faire le lien entre le binaire et l'hexadécimal. Mettre en évidence qu'un symbole hexadécimal correspond à 4 bits binaires.

- Passer en représentation hexadécimale.
- Mettre les 4 bits de poids le plus fort à 0.
- Demander aux élèves de changer comme ils veulent les 4 bits de poids le plus faible.
- Répéter l'opération cette fois en manipulant les 4 bits de poids le plus fort et en laissant les 4 bits de poids le plus faible à 0.
- Demander aux élèves ce qu'ils ont pu remarquer dans la représentation hexadécimale.

Activité 3 :

But : Montrer les limites de l'encodage 8 bits du bus de données et l'utilité d'utiliser un nombre accru de bits pour représenter de grands nombres.

- Demander aux élèves le plus grand nombre pouvant être représenté dans le bus de donnée (255).
- Idem dans le bus d'adresse (4095).

Activité 4 :

But : Utiliser le simulateur comme correcteur d'exercice de changement de base (ou comme aide).

- Version A : donner une liste d'exercices de changement de base aux élèves et leur laisser accès au simulateur pour trouver ou vérifier les réponses.
- Version B : Créer les exercices dans le simulateur en mémorisant les solutions en mémoire. Fournir le programme à l'élève qui peut le charger dans son simulateur et l'assembler. L'élève peut ensuite accéder à la réponse en sélectionnant l'adresse correspondant au numéro de l'exercice et en appuyant sur le bouton « Read ». Attention à écrire les valeurs dans l'assembleur dans la base de départ.

Exemple de programme :

```
MOVE #123, 0 ; place la valeur décimale 123 à l'adresse 0
MOVE #B'01101001, 1; place la valeur binaire 01101001 à l'adresse 1
MOVE #H'AB, 2; place la valeur hexadécimale AB à l'adresse 2
Halt
```