

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom :	Prénom :	Date :

1 Le Raspberry Pi

1.1 Présentation

Le Raspberry Pi est un nano ordinateur, c'est-à-dire un ordinateur de la taille d'une carte de crédit, coûtant environ 40 € et capable de diffuser des vidéos en 1080p (haute définition) à 30 images par seconde.

Nous réaliserons nos activités pratiques avec le Raspberry Pi 3.

Le Raspberry Pi 3 regroupe sur une carte de 85 × 56 mm tous les composants nécessaires pour faire tourner un système d'exploitation et utiliser ainsi le nano ordinateur comme un véritable PC.

Les outils de base sont disponibles dans la plupart des distributions : traitement de texte, tableur, jeux, navigateur Internet, outils de développement,....



1.2 Historique

Au début des années 2000, un petit groupe de personnes a l'ambition de rendre accessible au plus grand nombre de jeunes la programmation informatique. Pour satisfaire cette ambition, il fallait trouver une plateforme qui, à l'instar des anciens ordinateurs, pourrait démarrer directement dans un environnement de programmation. Pour contrer le fait qu'on ne peut pas laisser un jeune expérimenter ses programmes sur un PC, il faut une machine d'un prix tel que la notion de "risque" disparaisse.

À partir de 2008, les microprocesseurs conçus pour des utilisations mobiles sont devenus plus abordables, mais aussi suffisamment puissants pour fournir une vidéo d'excellente qualité. Le groupe a donc pensé que cette caractéristique pourrait rendre la carte attirante pour les jeunes, pas forcément intéressés par un outil destiné à la programmation pure.

Le projet semblait devenir réalisable. Eben Upton (aujourd'hui concepteur de composant chez Broadcom), Robert Mullins, Jack Lang et Alan Mycroft en équipe avec Pete Lomas, directeur de l'ingénierie chez Norcott Technologies, ainsi que David Braben, auteur du jeu Elite pour le BBC Micro, ont créé la Fondation Raspberry Pi pour que le projet devienne une réalité.

Trois ans plus tard, la fabrication en série commence, grâce à des accords de fabrication sous licence signés avec Farnell element14 et RS Components. Et ce n'est que le début de l'histoire du Raspberry Pi.

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom :	Prénom :	Date :

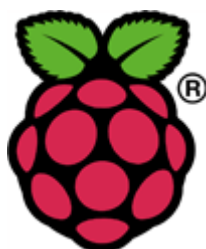
La Fondation Raspberry Pi, équivalent d'une association loi de 1901 en France, explique la genèse du projet Raspberry Pi sur son site web (www.raspberrypi.org/about).

1.3 Le logo

En dehors des considérations techniques et financières, la Fondation se préoccupait également de trouver une image pour la représenter. Il ne faisait aucun doute que le succès du Raspberry Pi lors de sa sortie allait avoir un retentissement dans les médias. Il fallait pouvoir identifier visuellement le produit.

Démarrée le 5 août 2011, la compétition posait comme contrainte que le logo devait être reconnaissable dans un carré de 1 cm par 1 cm, mais que sa qualité devait également permettre de l'utiliser sur différents produits de tailles différentes. Enfin, il devait être identifiable facilement aussi bien en couleur qu'en noir et blanc.

Le 7 octobre de la même année, Liz Upton publiait le résultat sur le site de la Fondation. Le logo dessiné par Paul Beech avait été retenu après le vote du jury.



Le choix de ce logo n'est pas innocent : il représente bien une framboise, mais sous la forme d'un fullerène C60, composé sphérique (ce qui rappelle le nombre soit 32 faces. Or le processeur du Raspberry Pi est un processeur 32 bits. De plus, sur ces 32 faces, 11 sont visibles sur le logo. Et le processeur du Raspberry Pi est un ARM... 11 ! Difficile de trouver plus de coïncidences.


il représente bien une framboise, mais sous la forme d'un fullerène C60, composé sphérique (ce qui rappelle le nombre soit 32 faces. Or le processeur du Raspberry Pi est un processeur 32 bits. De plus, sur ces 32 faces, 11 sont visibles sur le logo. Et le processeur du Raspberry Pi est un ARM... 11 ! Difficile de trouver plus de coïncidences.

Nous étudierons plus loin dans cette séquence, l'architecture matérielle du Raspberry Pi et de son processeur.

2 A la découverte de Raspbian

2.1 Qu'est-ce que c'est ?

Pour faire fonctionner un ordinateur, il faut un système d'exploitation. Cet OS (Operating System ou système d'exploitation) est le chef d'orchestre qui gère les ressources de l'ordinateur et les attribue aux diverses tâches en cours d'exécution, que ce soient des applications lancées par l'utilisateur ou des services propres au système lui-même.

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : _____ Prénom : _____		Date : _____



Lorsqu'un logiciel a besoin d'une ressource matérielle, c'est le système d'exploitation qui la lui attribue. L'OS gère le matériel (mémoire et périphériques), l'accès aux fichiers stockés sur les mémoires de masse, ainsi que l'échange d'informations sur le réseau.

Lorsque la Fondation a créé le Raspberry Pi, le maintien du prix sous la barre fixée au départ impliquait qu'il n'y ait pas de licence à payer pour l'utilisation du système d'exploitation. C'est donc tout naturellement que les créateurs se sont tournés vers Linux.

2.2 Linux

Linux devrait être appelé GNU/Linux. Il = GNU n'est pas Linux) lancé en 1983 d'exploitation libre sous licence GPL publique générale GNU). C'est-à-dire la disposition de tout un chacun. les codes, les étudier, les modifier et développeurs, parfois rémunérés par de Linux.



s'inscrit dans le projet GNU (GNU's NOT Linux par Richard Stallman. C'est un système (GNU General Public License = licence que les codes sources des programmes sont à N'importe qui peut utiliser Linux, reprendre les redistribuer. Ce sont des milliers de des entreprises, qui participent à l'évolution

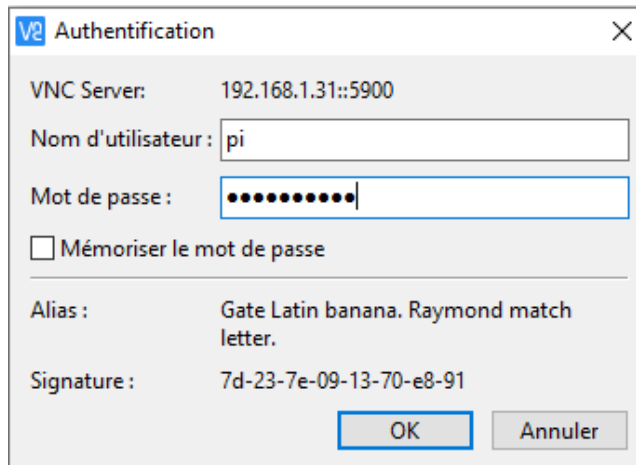
3 Découvrir l'OS Raspbian

Il existe deux modes d'utilisation du nano ordinateur Raspberry Pi.

3.1 Utilisation en mode graphique

A faire vous-même I

Connectez-vous à votre Raspberry Pi en utilisant les identifiants suivants :



V2 Authentification

VNC Server: 192.168.1.31::5900

Nom d'utilisateur: pi

Mot de passe: [masked]


☐ Mémoriser le mot de passe

Alias: Gate Latin banana. Raymond match letter.

Signature: 7d-23-7e-09-13-70-e8-91

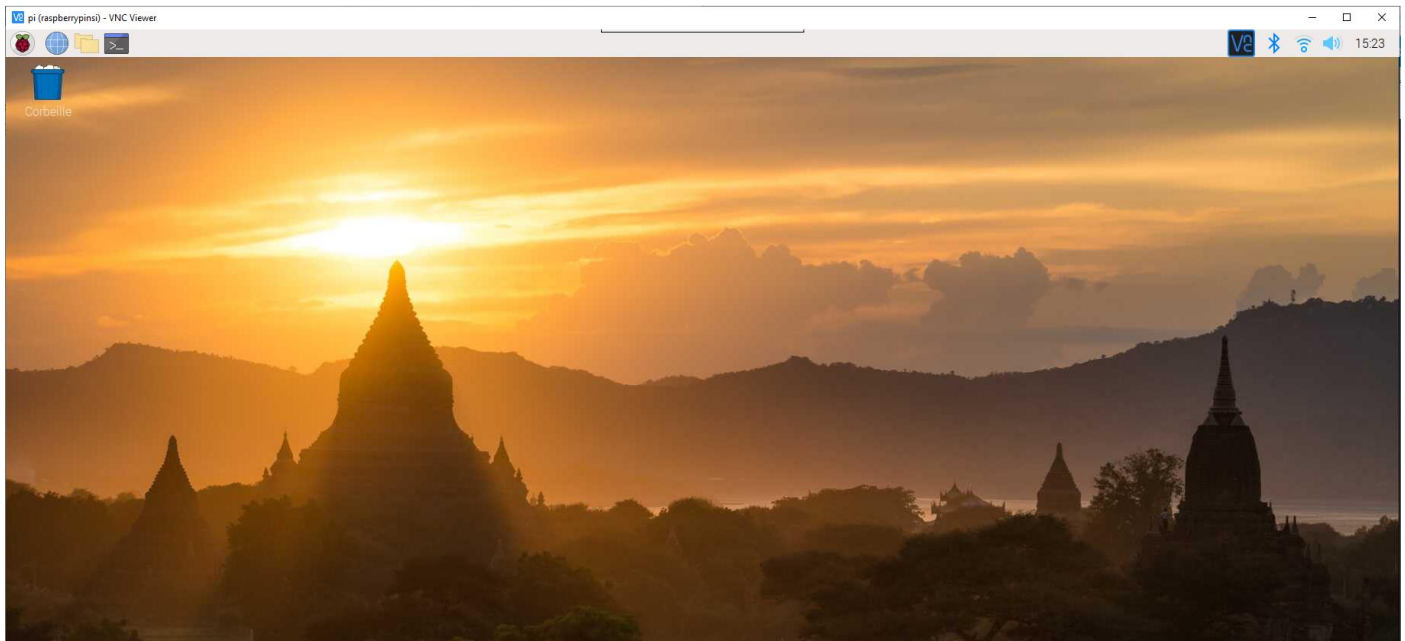
OK Annuler

Le mdp par défaut est « raspberrypi ».

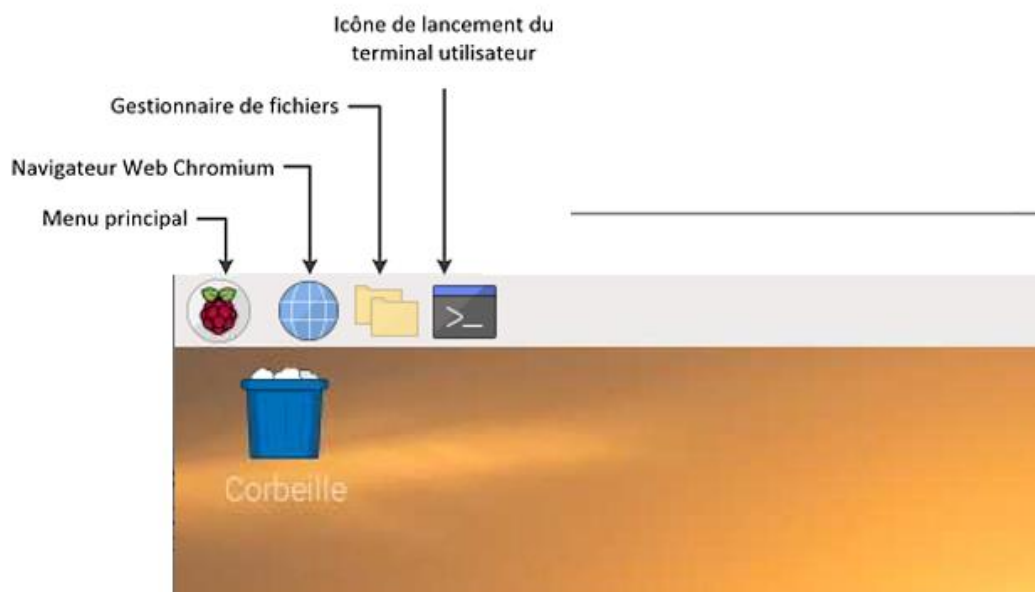
Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : _____ Prénom : _____		Date : _____

Victor Hugo

Une fois connecté, vous devez obtenir l'écran d'accueil « Raspbian » suivant:



La barre des tâches comporte une partie gauche (ci-dessous), une zone centrale affichant les fenêtres ouvertes sur le bureau ou mises en réduction et une partie droite donnant des informations sur les connexions réseau actives ou encore l'heure.



Prenez un peu de temps pour découvrir les menus offerts par la barre de tâches.

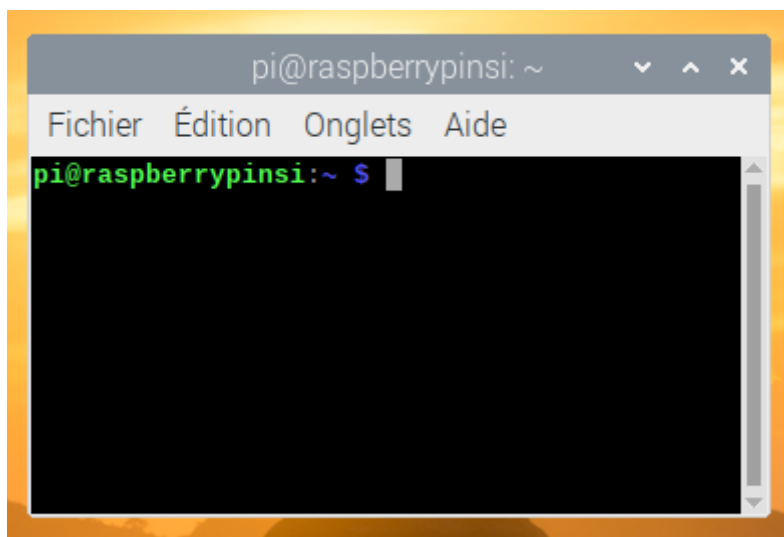
Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : _____ Prénom : _____		Date : _____

3.2 Utilisation de la ligne de commande

L'utilisation du mode « texte » est certes moins convivial mais il permet une meilleure compréhension du fonctionnement du système d'exploitation « Raspbian ». C'est par conséquent le mode que nous allons privilégier dans cette activité.

A faire vous-même II

- a. Ouvrez une fenêtre « Terminal de commande » en cliquant sur l'icône « LXTerminal » :



Le système de fichiers de Raspbian (Linux) est organisé à partir d'un point de départ appelé « root », racine ou encore /. Sous cette racine se déploient des répertoires contenant les fichiers et programmes nécessaires au système d'exploitation.

- A tout moment, situez-vous dans cette arborescence en tapant la commande « pwd »
- Remontez au plus haut niveau de l'arborescence en tapant la commande « cd / » (cd pour « change working directory »)
- Découvrez le nom des répertoires (« directories ») situés à la racine en tapant la commande « ls »
- Combien de répertoires figurent à la racine de l'arborescence ?

- f. Visualisez l'arborescence de votre système avec la commande « tree -L 1 »



Architecture matérielles
& Systèmes d'exploitation

A la découverte d'un Nano Ordinateur



Activité Pratique

Nom :

Prénom :

Date :

- g. Atteignez le répertoire /home avec la commande « cd home ». Quelle information vous précise que vous êtes bien dans le répertoire /home ?


- h. Entrer à nouveau la commande « ls ». Quel résultat obtenez-vous ?



A retenir : le contenu des principaux répertoires de l'arborescence « Linux »

/	Racine ou root, contient les répertoires de l'arborescence Linux.
bin	Exécutables binaires du système <i>cp, ls, mount, rm...</i>
boot	Fichiers de démarrage de Linux.
dev	Fichiers spéciaux assurant la liaison avec les périphériques.
etc	Fichiers de configuration du système, des services...
home	Répertoire personnel des utilisateurs.
lib	Bibliothèques système partagées.
media	Point de montage des clés USB, CD-ROM...
mnt	Point de montage temporaire de partitions et périphériques.
proc	Informations sur les processus et le noyau Linux.
root	Répertoire personnel du super-utilisateur.
sbin	Binaires système et outils comme <i>fsck</i> .
tmp	Fichiers temporaires.
usr	Fichiers binaires et commandes utilisateurs.
var	Système de fichiers "variables" (modifiables) ; on y trouve le contenu web (répertoire <i>www</i>), mais aussi les logs (journaux).

Ces répertoires contiennent eux-mêmes d'autres répertoires qui constituent l'arborescence de Linux. Quelques différences peuvent exister entre les distributions, mais la plus grande partie de l'arborescence est commune.

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Victor Hugo		
Nom :	Prénom :	Date :

A faire vous-même III

Nous allons apprendre à :

- Lister le contenu d'un répertoire
 - Nous déplacer dans l'arborescence
- a. Positionnez-vous dans le répertoire « etc » en tapant la commande « cd /etc »
 - b. Taper la commande « ls -al »




A retenir : la commande « ls »

La commande ls (list sort = liste triée) affiche le contenu d'un répertoire : fichiers et sous-répertoires. Cette commande supporte de nombreuses options, en particulier des options de tri pour la présentation du résultat. Seules les plus utiles sont listées ci-dessous :

-F	Ajoute un / à la fin du nom des répertoires.
-R	Affiche récursivement le contenu des sous-répertoires. Si un sous répertoire existe, son contenu est affiché.
-l	Affiche des informations très complètes sur les fichiers et répertoires (type, droits d'accès, nom du propriétaire...).
-a	Affiche tous les fichiers, y compris ceux qui commencent par un . et qui sont des fichiers cachés.
nom_de_fichier	Nom de fichier à afficher, accepte les "jokers" * et ?.

Pour connaître toutes les options de la commande, taper la commande « man ls »

- c. Taper la commande « ls -al | more ». Enfoncez la barre d'espace pour activer le défilement page par page. Que constatez-vous ?
- d. Selon vous, « vim » est-il un fichier ou bien un répertoire ? Comment pouvez-vous le vérifier simplement ?

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : _____ Prénom : _____		Date : _____



- e. Déplacez-vous dans le répertoire « xml »
- f. Depuis le répertoire « xml », déplacez-vous dans le répertoire « /etc/vim ». Pour cela, vous avez deux possibilités :
 1. Utiliser un chemin « absolu » en tapant la commande « cd /etc/vim »
 2. Utiliser un chemin « relatif » en tapant la commande « cd ../vim ». Au préalable, positionnez-vous dans le répertoire « xml »
 Testez les deux possibilités.
- g. Taper la commande « cd \$HOME ». Quel son effet ?



A retenir : chemin absolu – chemin relatif

Il existe deux façons de se déplacer dans l'arborescence : le mode absolu, qui repart systématiquement de la racine, et le mode relatif, qui part du répertoire de travail actif.

Les deux méthodes de repérage dans l'arborescence ont des utilisations différentes. En absolu, on est sûr de toujours arriver dans le bon répertoire. L'adressage relatif est souvent utilisé dans les sites web, ce qui permet de déplacer une partie de l'arborescence dans laquelle on a utilisé l'adressage relatif, sans conséquence pour les liens entre les pages (cf. « L'interaction Homme-Machine »).

A faire vous-même IV

Nous allons apprendre à modifier l'arborescence Linux

- a. Déplacez-vous dans le répertoire « /home/pi » avec la commande « cd \$HOME »
- b. Notez le contenu du répertoire ci-après :

.....

.....

.....

.....

- c. Créez le répertoire « Mes_doc_NSI » en tapant la commande « mkdir Mes_doc_NSI »
- d. Vérifiez l'effet de la commande précédente avec la commande « ls -l ». La date de création du répertoire est-elle bien celle du jour ?
- e. Déplacez-vous dans le répertoire « /home/pi/ Mes_doc_NSI »

<div>Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS</div> <div></div>		<div>NSI</div> <div>NUMÉRIQUE ET SCIENCES INFORMATIQUES</div>
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : Prénom : Date :		

f. Créez les répertoires :

- Mes_photos
- Mes_doc_pdf
- Mes_sites_web

g. Taper la commande « `mkdir -p Mes_algorithmes/programmes` ». Quel a été son effet ?

A faire vous-même V

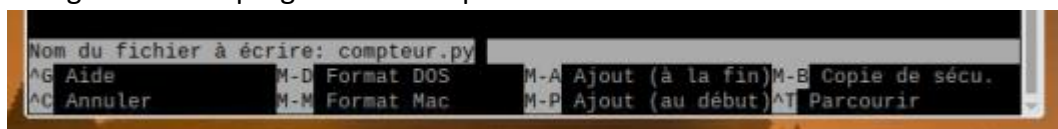
Nous allons apprendre à créer un fichier.

A Avec un éditeur de texte :


- Déplacez-vous dans le répertoire « `/home/pi/ Mes_doc_NSI/ Mes_algorithmes/programmes` » en utilisant un chemin relatif
- Tapez la commande « `nano compteur.py` ». Vous parvenez alors dans la fenêtre de l'éditeur de texte « nano ». nano est un éditeur de texte tout simple. Voici les commandes essentielles :



- Ecrire un programme en langage Python qui initialise un compteur qui compte et affiche les entiers de 0 à 25 inclus. Faites attention à l'encodage, il est de type « ANSI ».
- Enregistrez votre programme en tapant sur « Entrée »



- Vérifiez que votre fichier a été créé dans le répertoire « programmes »
- En ligne de commande, exécutez votre programme avec la commande « `python3 compteur.py` »

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Victor Hugo		
Nom :	Prénom :	Date :

B En ligne de commande :

g. Exécuter la commande « touch vide1 ». Quel est son effet ?

.....nom du fichier créé :Date de création :taille :

h. Puis tapez la commande « man echo »

i. Servez-vous de la commande « echo » afin que le fichier « vide1 » contienne la chaîne de caractères : « Bonjour les NSI ! ». Quelle est la taille du fichier « vide1 » ?

j. Puis tapez la commande « mv vide1 Bonjour ». Quel est son effet ?

.....

k. Vérifiez le contenu du fichier « Bonjour » avec l'éditeur « nano ».

A faire vous-même VI

Nous allons apprendre à copier. La copie de fichier et de répertoire sous Linux se fait avec la commande cp. Cette commande accepte plusieurs syntaxes.


a. Depuis le répertoire courant « programmes », créez le répertoire « /home/pi/ Mes_doc_NSI/ Mes_algorithmes/programmes_python » en utilisant un chemin relatif.

b. Tapez la commande :

```
pi@raspberrypi:~/Mes_doc_NSI/Mes_algorithmes/programmes $ cp compteur.py ../programmes_python/compteur2.py
```

c. Quel est son effet ?

.....

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Victor Hugo		
Nom :	Prénom :	Date :

d. Positionnez-vous dans le bon répertoire afin de pouvoir modifier le programme compteur2.py selon le résultat suivant :

```
compteur vaut: 0
compteur vaut: 1
compteur vaut: 2
compteur vaut: 3
compteur vaut: 4
compteur vaut: 5
compteur vaut: 6
compteur vaut: 7
compteur vaut: 8
compteur vaut: 9
compteur vaut: 10
compteur vaut: 11
compteur vaut: 12
compteur vaut: 13
compteur vaut: 11
compteur vaut: 10
compteur vaut: 9
compteur vaut: 8
compteur vaut: 7
compteur vaut: 6
compteur vaut: 5
compteur vaut: 4
compteur vaut: 3
compteur vaut: 2
compteur vaut: 1
compteur vaut: 0
```



Copier un fichier vers un autre fichier. Si le fichier n'existe pas, le fichier est copié. Si la destination existe, le fichier est écrasé (voir l'option -i ci-dessous).

Syntaxe d: cp [-iu] fichier destination

fichier	Nom du fichier à copier, appelé aussi fichier source.
destination	Si destination est un nom de fichier, le fichier source est recopié dans le fichier destination. Si destination est un nom de répertoire, le fichier source est copié dans ce répertoire.
-i	Cette option indique au système d'interroger l'utilisateur avant d'écraser un fichier existant.
-u	Le système ne fait pas la copie si le fichier destination a une date de dernière modification égale ou plus récente que celle du fichier source.



Architecture matérielles
& Systèmes d'exploitation

A la découverte d'un Nano Ordinateur



Activité Pratique

Nom :

Prénom :

Date :



e. Copier plusieurs fichiers vers un répertoire

Syntaxe : `cp [-iu] fichier_1 fichier_2... fichier_n répertoire`

fichier_n	Les fichiers 1 à n sont les fichiers à copier.
répertoire	Le répertoire dans lequel les fichiers seront copiés.
-i et -u	Ces options ont le même effet que dans la syntaxe d.

Application : en vous positionnant dans le bon répertoire, copier les « Bonjour » et « compteur.py » dans le répertoire « /home/pi/ Mes_doc_NSI/ Mes_algorithmes/programmes_python ». Vérifiez le succès de l'opération.



f. Copier un répertoire et ses sous-répertoires vers un répertoire.

Syntaxe : `cp [-iuR] répertoire_source répertoire_destination`

répertoire_source	Répertoire à copier.
répertoire_destination	Répertoire dans lequel le répertoire source et ses sous-répertoires seront copiés.
-i et -u	Ces options ont le même effet que dans la syntaxe d.
-R	Copie récursivement les sous-répertoires du répertoire source. La présence de -R lors de la copie de répertoires évite l'affichage d'un message d'erreur.

Application :

1. Au sein du répertoire « /home/pi/ Mes_doc_NSI », créez le répertoire « /Mes_programmes »
2. Copiez le répertoire « /Mes_algorithmes » ainsi que son arborescence vers le répertoire « /home/pi/ Mes_doc_NSI/Mes_programmes »
3. Vérifiez la bonne exécution de l'opération en utilisant une seule commande.

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom :		Prénom :
		Date :


Votre arborescence ressemble maintenant à :

```

├── Desktop
├── Documents
├── Downloads
│   └── afvm8.html
├── MagPi
│   └── MagPi83.pdf
├── Mes_doc_NSI
│   ├── Mes_algorithmes
│   │   ├── programmes
│   │   │   ├── Bonjour
│   │   │   └── compteur.py
│   │   └── programmes_python
│   │       ├── Bonjour
│   │       ├── compteur2.py
│   │       └── compteur.py
│   ├── Mes_doc
│   ├── Mes_notes
│   ├── Mes_programmes
│   │   ├── Mes_algorithmes
│   │   │   ├── programmes
│   │   │   └── programmes_python
│   └── Mes_sites_web
├── Music
├── Pictures
│   └── nsi.jpg
├── Public
├── Templates
└── Videos

20 directories, 8 files
pi@rasberrypinsi:~ $

```

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Victor Hugo		
Nom :	Prénom :	Date :

A faire vous-même VII

Nous allons maintenant apprendre à déplacer et renommer les fichiers.

La commande mv (*move* = déplacer) permet de déplacer mais aussi de renommer fichiers et répertoires.

Sa syntaxe est : mv [-fiuv] source destination

source	Nom du fichier ou du répertoire source.
destination	Nom du fichier ou du répertoire destination. Si destination est un nom de répertoire, mv déplace les fichiers sources dans ce répertoire. Si source et destination sont des fichiers dans le même système de fichiers, mv renomme le fichier. Si source et destination sont des fichiers dans des systèmes de fichiers différents, mv déplace le fichier. Si source est un nom de répertoire, mv renomme ce répertoire.
-f	Force l'écrasement des fichiers existants.
-i	Interroge l'utilisateur avant d'écraser un fichier. Si -f et -i existent sur la ligne de commande, c'est le dernier qui l'emporte.
-u	Le système ne fait pas la copie si le fichier destination a une date de dernière modification égale ou plus récente que celle du fichier source.
-v	Indique le nom des fichiers déplacés sur l'écran.

Application :

- Positionnez-vous dans le répertoire « /home/pi/ Mes_doc_NSI/Mes_programmes »
- Déplacez le répertoire « /home/pi/ Mes_doc_NSI/Mes_programmes/Mes_algorithmes/programmes_python » vers « /home/pi/ Mes_doc_NSI/Mes_programmes » en le renommant « /home/pi/ Mes_doc_NSI/Mes_programmes/Mes_programmes_python »
- Vérifiez la bonne exécution de l'opération en utilisant une seule commande.
- Positionnez-vous dans « /home/pi/ Mes_doc_NSI/Mes_programmes/Mes_programmes_python »
- Renommez « compteur.py » en « compteur0_25.py » et « compteur2.py » en « compteur0_13_0.py »
- Depuis le répertoire « /home/pi/ Mes_doc_NSI/Mes_programmes/Mes_programmes_python », déplacez le fichier « Bonjour » vers le répertoire « /home/pi/ Mes_doc_NSI/Mes_doc » en utilisant un chemin relatif.


<div>Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS</div> <div></div> <div><div>NSI</div><div>NUMÉRIQUE ET SCIENCES INFORMATIQUES</div></div>		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : Prénom : Date :		

Votre arborescence ressemble maintenant à :

```

— Desktop
— Documents
— Downloads
  └─ afvm8.html
— MagPi
  └─ MagPi83.pdf
— Mes_doc_NSI
  └─ Mes_algorithmes
    └─ programmes
      └─ Bonjour
      └─ compteur.py
    └─ programmes_python
      └─ Bonjour
      └─ compteur2.py
      └─ compteur.py
  └─ Mes_doc
    └─ Bonjour
  └─ Mes_notes
  └─ Mes_programmes
    └─ Mes_algorithmes
      └─ programmes
    └─ Mes_programmes_python
      └─ compteur0_13_0.py
      └─ compteur0_25.py
  └─ Mes_sites_web
— Music
— Pictures
  └─ nsi.jpg
— Public
— Templates
— Videos

```

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : _____ Prénom : _____ Date : _____		

A faire vous-même VIII

Nous allons maintenant apprendre à supprimer les fichiers.

La suppression de fichier se fait avec la commande `rm` (*remove* = supprimer).

Sa syntaxe est : `rm [-fiRv] nom_de_fichier`

nom_de_fichier	Nom du fichier ou du répertoire à effacer.
-f	Ne pas avertir l'utilisateur si un fichier à effacer n'existe pas. Forcer l'effacement sans demander de confirmation si le fichier existe.
-i	Interroge l'utilisateur avant d'effacer un fichier. Si -f et -i existent sur la ligne de commande, c'est le dernier qui l'emporte.
-R	Efface récursivement les sous-répertoires du répertoire source et le répertoire lui-même.
-v	Indique le nom des fichiers effacés sur l'écran.

Application : nettoyage de notre arborescence

- Supprimez le répertoire « `/home/pi/ Mes_doc_NSI/ Mes_algorithmes` » avec son contenu
- Supprimez le répertoire « `/home/pi/ Mes_doc_NSI/Mes_programmes/Mes_algorithmes` » avec son contenu
- Supprimez le répertoire « `/home/pi/ Mes_doc_NSI/Mes_programmes/Mes_algorithmes/programmes` » avec son contenu
- Vérifiez le contenu de votre arborescence. Celle-ci doit être similaire à :

```

pi@raspberrypinsi: ~
Fichier  Édition  Onglets  Aide

Desktop
Documents
Downloads
├── afvm8.html
├── MagPi
│   └── MagPi83.pdf
├── Mes_doc_NSI
│   ├── Mes_doc
│   │   └── Bonjour
│   ├── Mes_notes
│   ├── Mes_programmes
│   │   ├── Mes_algorithmes
│   │   └── Mes_programmes_python
│   │       ├── compteur0_13_0.py
│   │       └── compteur0_25.py
│   └── Mes_sites_web
├── Music
├── Pictures
│   └── nsi.jpg
├── Public
├── Templates
└── Videos
  
```

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS			
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur		
	Activité Pratique		
Nom : _____		Prénom : _____	Date : _____

A faire vous-même IX



Nous allons maintenant apprendre à afficher le contenu d'un fichier. L'affichage du contenu d'un fichier se fait avec la commande cat.

Sa syntaxe est : `cat [-n] nom_de_fichier_1 nom_de_fichier_2`

nom_de_fichier_1	Nom du fichier dont l'utilisateur veut visualiser le contenu. Si plusieurs noms de fichiers sont présents dans la ligne de commande, cat affiche leur contenu à l'écran à la suite.
-n	Numérote les lignes affichées sur l'écran. Utile quand il faut retrouver une ligne qui pose problème.

Application :

- Afficher le contenu des deux programmes « `compteur0_25.py` » et « `compteur0_13_0.py` » avec leurs numéros de ligne.
- Editer les fichiers afin de mettre à jour le libellé de chaque programme

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : _____ Prénom : _____		Date : _____

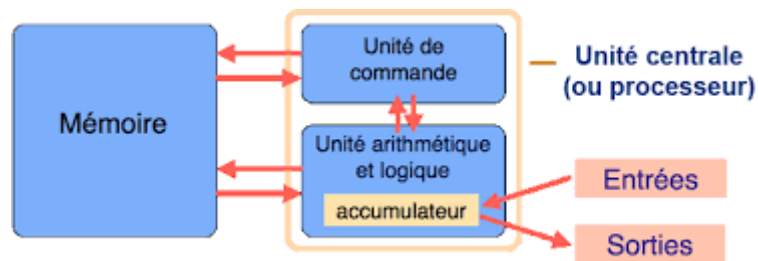
4 Qu'est-ce qu'une Architecture matérielle ?

4.1 Une approche globale

Il existe plusieurs échelles de description du fonctionnement d'un ordinateur. Son architecture matérielle est l'une d'elles. Même si nous notre analyse n'est pas très poussée, elle va nous permettre d'en comprendre les principes d'organisation.

Selon l'encyclopédie Wikipédia, l'architecture matérielle décrit l'agencement interne de composants électroniques ainsi que leurs interactions.


Un ordinateur est principalement composé de deux grands circuits : le processeur et la mémoire. Ces deux circuits sont reliés entre eux par des fils qui constituent un ou plusieurs bus de communication. Parmi ceux-ci, nous distinguons un bus de données et un bus d'adresses. L'unité centrale (ou processeur) est composée de deux unités. L'unité de contrôle lit en mémoire un programme et donne à l'unité arithmétique et logique (unité de calcul) la séquence des instructions à effectuer. Le processeur dispose par ailleurs de bus d'entrées et de sorties permettant d'accéder aux autres parties de l'ordinateur, que l'on nomme les périphériques. Cette organisation générale porte le nom d'architecture de Von Neumann.



Cette architecture doit son nom au mathématicien Hongrois John Von Neumann. En 1944, John Von Neumann est introduit dans le projet ENIAC par Herman Goldstine, qui assurait la liaison scientifique du projet avec le département de la Défense. Von Neumann était un esprit universel, dont les contributions allaient des mathématiques à la logique, la physique et l'économie. Il avait rencontré Turing et connaissait ses travaux.

L'ENIAC (*Electronic Numerical Integrator and Computer*) fut le premier calculateur entièrement électronique ayant les mêmes capacités qu'une machine de Turing, aux limites physiques près.

A l'aide d'Internet, définissez et expliquez le fonctionnement de la machine de Turing

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : _____		Prénom : _____
		Date : _____

4.2 Principes de fonctionnement

La mémoire

Elle est composée de plusieurs milliards de circuits mémoires un bit. Ces circuits sont organisés en agrégats de huit, seize, trente-deux, soixante-quatre bits, et parfois davantage, que l'on appelle des cases mémoires et qui peuvent donc mémoriser des mots de huit, seize, trente-deux, soixante-quatre bits, etc. Le nombre de ces cases définit la taille de la mémoire de l'ordinateur. Comme il faut distinguer ces cases les unes des autres, on donne à chacune un numéro : son adresse. La mémoire contient les données sur lesquelles on calcule et le programme qui décrit le calcul effectué, donné sous la forme d'une séquence d'instructions.

Le processeur




Il n'a qu'un très petit nombre de cases mémoires, que l'on appelle des registres. On peut imaginer, par exemple, qu'il ne contient que quelques registres. Les registres peuvent contenir des données, mais aussi des adresses de cases mémoires. Lorsque l'on parle de processeurs 32 bits ou 64 bits, on fait référence à la taille de ces registres.

Communication Mémoire – Processeur

Pour échanger des données avec la mémoire, le processeur utilise des procédés qui permettent :

- de transférer l'état d'un registre dans une case mémoire
- de transférer l'état d'une case mémoire dans un registre

Pour transférer le contenu d'un registre $r0$ dans la case mémoire d'adresse n , le processeur met les différents fils qui composent le bus d'adresses dans un état qui correspond à l'expression en base deux du nombre n et il met les différents fils qui composent le bus de données dans un état qui correspond au contenu du registre. Au signal d'horloge, chaque case de la mémoire compare son propre numéro au numéro arrivé sur le bus d'adresse ; seule la case numéro n se reconnaît et elle met alors ses différentes entrées dans l'état 1, de manière à enregistrer le mot arrivant sur le bus de données. Le procédé symétrique permet au processeur de récupérer une valeur précédemment enregistrée : les informations circulent toujours du processeur vers la mémoire sur le bus d'adresses, mais elles circulent dans l'autre sens sur le bus de données : c'est la case n qui connecte sa sortie au bus de données et c'est le registre qui met ses entrées à 1 de manière à enregistrer le mot qui arrive sur le bus de données. Ces deux opérations s'appellent le stockage et le chargement du contenu d'une case mémoire dans le registre $r0$.

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom :	Prénom :	Date :

5 Un exemple d'architecture : le Raspberry Pi

5.1 Présentation

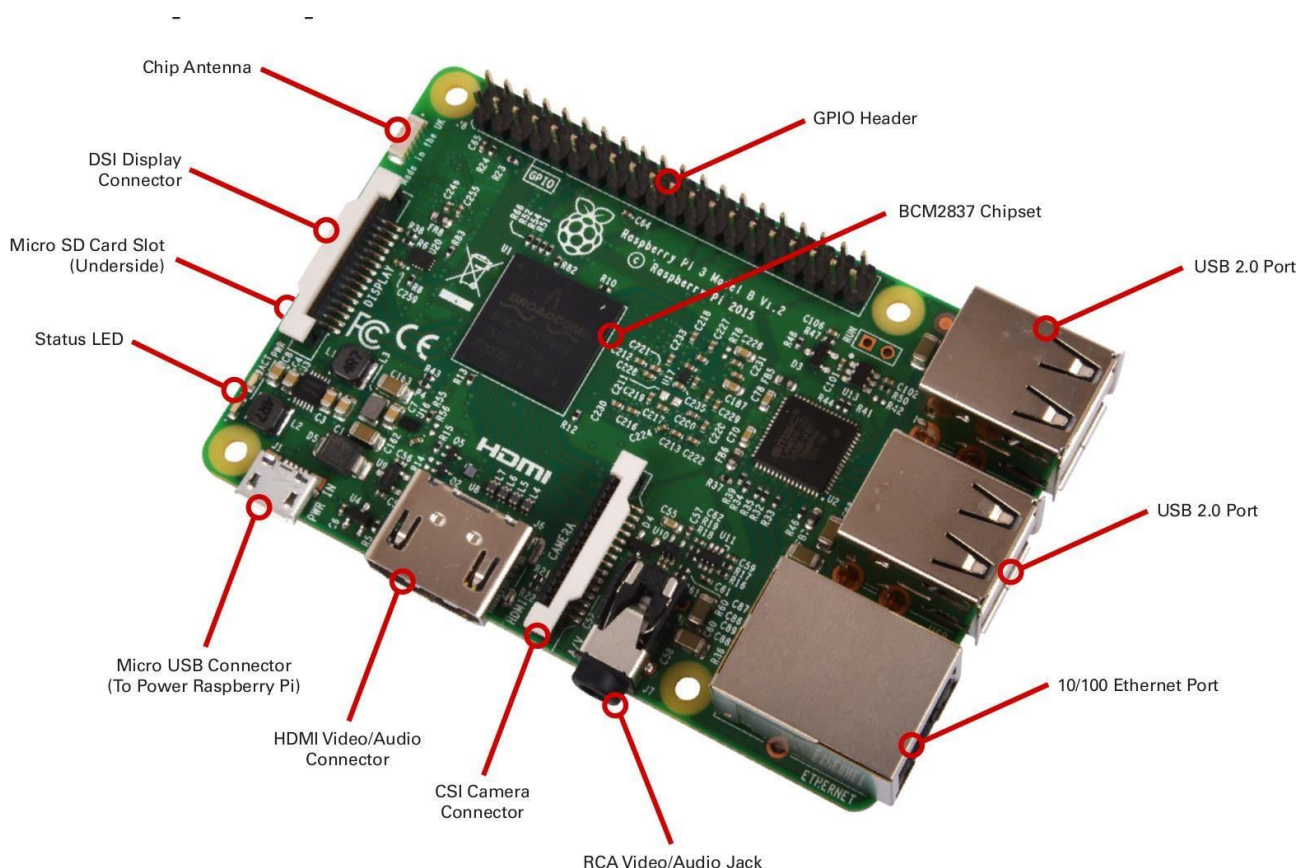
Le Raspberry Pi est un nano-ordinateur mono carte à processeur ARM conçu par des professeurs du département informatique de l'université de Cambridge dans le cadre de la fondation Raspberry Pi.



Ce nano ordinateur, qui a la taille d'une carte de crédit, permet l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux (notamment Debian) et des logiciels compatibles. Mais il fonctionne également avec l'OS Microsoft Windows : Windows 10 IoT Core3 et celui de Google, Android Pi.

5.2 Architectures matérielles

La carte possède :

- Un processeur (ARM à 700 MHz ou 1,2 GHz).
- 1, 2 ou 4 ports USB, un port RJ45
- 256 Mo (ou 1Go) de mémoire vive.
- Son circuit graphique BMC VideoCore 4 permet de décoder des flux Blu-Ray full HD (1080p 30 images par seconde), d'émuler d'anciennes consoles et d'exécuter des jeux vidéo relativement récents



Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom :	Prénom :	Date :

5.3 La carte Raspberry Pi 3 Modèle B

Nous allons baser notre étude sur la carte La carte Raspberry Pi 3 Modèle B.

Celle-ci est équipée d'un Quad-Core ARM Cortex-A53 1.2 GHz (Broadcom BCM2837), de 1024 Mo de RAM et d'un GPU Dual-Core VideoCore IV capable de décoder les flux vidéo HD 1080p.

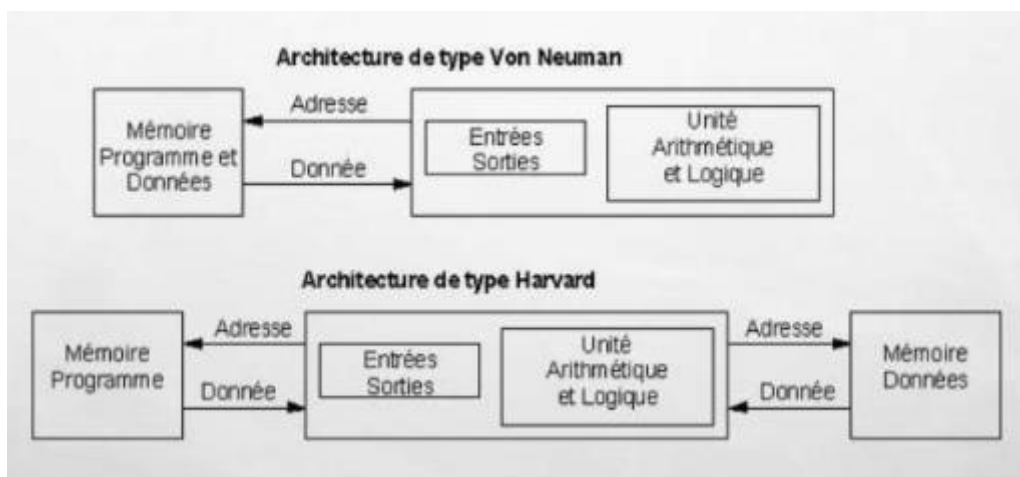
Ci-dessous figure la liste des principaux connecteurs :

- Carte mère Raspberry Pi 3 Type B
- Processeur intégré Quad-core ARM Cortex-A53 1.2 GHz (Broadcom BCM2837)
- RAM : 1024 Mo
- GPU Dual Core VideoCore IV Multimedia Co-Processor
- Lecteur de cartes Micro SD
- Ports : HDMI, 4x USB, RJ45, jack 3.5 mm, connecteurs pour APN et écran tactile
- Connecteur GPIO 26 broches
- Wi-Fi b/g/n et Bluetooth 4.1
- Support des distributions dédiées basées sur Linux et Windows 10

5.4 Le chipset ARM

Les architectures ARM sont des architectures externes de type RISC 32 bits (ARMv1 à ARMv7) et 64 bits (ARMv8) développées par ARM Ltd depuis 1983 et introduites à partir de 1990 par Acorn Computers.

Cette famille de processeurs est dotée d'une architecture de type « Harvard ».



Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : _____ Prénom : _____		Date : _____



A partir du schéma d'architecture ci-dessus, notez les différences notables entre les deux modèles d'architecture, « Von Neumann » et « Harvard » :

5.5 Jeu d'instructions, langage Machine et Assembleur

ARM est une famille d'architectures qui comprend des jeux d'instructions pour processeurs informatiques. Le langage machine est construit à partir d'instructions implémentées pour un processeur particulier.



Afin de vérifier les informations concernant le processeur de votre carte Raspberry Pi, afficher le contenu du fichier « /proc/cpuinfo »

Le langage machine est interprété par le processeur en termes de codes binaires. Le code binaire est ce qu'un ordinateur peut exécuter. Il est composé d'instructions codées dans une représentation binaire (ces codages sont documentés dans les manuels ARM).

Vous pourriez écrire des instructions de codage en code binaire, mais ce serait fastidieux. Nous allons donc écrire dans un langage qui s'appelle le **langage assembleur**. Le langage assembleur est juste une couche de syntaxe mince au-dessus du code binaire. Apprendre les rudiments de l'assembleur est un bon moyen de comprendre le fonctionnement d'un processeur.



Nous utiliserons donc un outil (GNU Assembler) pour assembler le code assembleur en un code binaire que nous pourrions exécuter.

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom :	Prénom :	Date :

5.6 Programmation en Assembleur

Notre objectif est de découvrir et de comprendre le fonctionnement d'un processeur. Il s'agit pour nous de soulever le capot de la « machine » et de ne plus considérer celle-ci comme une boîte noire. Nous ne serons donc pas trop ambitieux !

A faire vous-même I : Notre premier programme

- Créez le répertoire « /home/pi/ Mes_doc_NSI/Mes_programmes/Mes_programmes_assembleur »
- Puis ouvrez l'éditeur Geany :



- Saisissez le code suivant :

```

1  /* Assembleur ARM raspberry Pi 3 modèle B */
2  /* pour vérification compilation et link */
3
4  .global main /* point d'entrée du programme */
5
6  main: /* programme principal */
7      mov r0,#4 /* on met 4 dans le registre r0 qui sera le code retour du programme */
8      bx lr /* fin du programme par retour à l'adresse contenu dans lr */

```


- Sauvegardez votre programme sous « /home/pi/ Mes_doc_NSI/Mes_programmes/Mes_programmes_assembleur » avec le libellé « programme_1.s »
- Puis assemblons (on dit aussi compiler) le programme avec la commande :

```

pi@raspberrypi:~ $ as -o programme_1.o programme_1.s

```

Le fichier « programme_1.o » est le résultat de l'assemblage (la compilation) du fichier « programme_1.s ». C'est un fichier objet.

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : _____ Prénom : _____		Date : _____



f. Afficher le contenu du répertoire. Que constatez-vous ?

g. Editer le fichier « programme_1.o » avec l'éditeur hexadécimal hexedit :

```
pi@raspberrypi:~/Mes_doc_NSI/Mes_programmes/Mes_programmes assembleur $ hexedit programme_1.o
```

Interpréter le résultat obtenu ci-dessous :


h. Et enfin, éditons les liens du programme (linkage en anglais) avec la commande :

```
pi@raspberrypi:~ $ gcc -o programme_1 programme_1.o
```

i. Editer le contenu du répertoire. Que constatez-vous ?

j. Taper la commande : \$ file programme_1. Quel est le type du fichier « programme_1 » ?

k. Lancez programme_1 avec la commande « ./programme_1 ». Que s'est-il passé ? Pourquoi ?

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : _____		Prénom : _____
		Date : _____



I. Taper la commande « echo \$? ». Que constatez-vous ?

Quelques explications...pour comprendre

Comme annoncé, nous commençons avec un programme simple qui ne fait que renvoyer un code. On interprète ce type de retour comme une valeur qui nous donne la nature d'une erreur.

Nous avons écrit une fonction principale en assembleur qui retourne la valeur « 4 ». Pour simplifier un peu plus les choses, le programme est « encapsulé » par un environnement d'exécution (runtime C) qui gère l'initialisation et la terminaison du programme pour nous.

Comme avec le langage C, les commentaires sont inclus entre /* ... */. Nous les utilisons pour documenter notre assembleur car tout ce qui apparait entre /* ... */ est ignoré.

Expliquons le sens de chaque instruction du programme :

```
.global main /* point d'entrée du programme */
```

Ceci est une directive pour le logiciel GNU Assembler. Une directive indique au logiciel de faire quelque chose de spécial. Les directives commencent par un point (.) suivi du nom de la directive et de certains arguments.

Ci-dessus, nous déclarons que « main » est le point d'entrée du programme. Or l'environnement d'exécution appellera « main ». Si ce n'est pas déclaré en « global », ce n'est pas appelable par l'environnement d'exécution et la phase d'édition de liens échoue.

```
main: /* programme principal */
```

Chaque ligne de GNU Assembler qui n'est pas une directive sera toujours de la forme « label: instruction ». Ici, « main : » seul est interprété comme une étiquette qui marque le début du corps du programme. Tout ce qui vient après doit être indenté afin d'appartenir au corps du programme.

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom :	Prénom :	Date :

Puis viennent deux instructions d'assembleur GNU :

1.

```
mov r0,#4
```

L'indentation suggère visuellement que cette instruction appartient au corps du programme. Ici, l'instruction « mov » déplace la valeur « 4 » dans le registre r0. La syntaxe est particulière car la destination est à gauche.

2.

```
bx lr
```

Un processeur ARM exécute les instructions séquentiellement (cette exécution séquentielle n'est pas spécifique à ARM et a lieu dans presque toutes les architectures).


Par conséquent, après l'instruction « mov », l'instruction « bx » sera exécutée. Cette instruction bx signifie littéralement « branchement et échange » (pour l'instant, ne nous préoccupons pas de l'échange). Le branchement est utilisé pour modifier le flux d'exécution des instructions.

Dans ce cas, nous nous connectons directement au registre « lr ». Nous ne nous soucions pas maintenant de ce que le registre « lr » contient. Il suffit de comprendre que cette instruction met ainsi fin au programme. Le résultat de main est le code d'erreur du programme qui est retourné à l'environnement d'exécution. Lorsque vous quittez le corps du programme, ce résultat est stocké dans le registre r0.

Les Registres

À la base, un processeur dans un ordinateur n'est autre chose qu'une calculatrice puissante. Les calculs ne peuvent être effectués qu'à l'aide de valeurs stockées dans de toutes petites mémoires appelées « registres ». Le processeur ARM d'un Raspberry Pi possède 16 registres d'entiers et 32 registres de réels en notation « virgule flottante » (cf. La représentation des données : représentation des nombres à virgule). Un processeur utilise ces registres pour effectuer des calculs de nombres entiers et des calculs de virgule flottante, respectivement.

Les 16 registres d'entiers d'un processeur ARM sont accessibles en utilisant les libellés r0 à r15. Ils peuvent contenir 32 bits. Bien sûr, ces 32 bits peuvent encoder ce que vous voulez. Cela dit, il est commode de représenter des entiers en complément à deux (cà2, cf. La représentation des données : représentation des entiers relatifs) car il existe des instructions qui effectuent des calculs en utilisant ce codage par défaut.

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom :	Prénom :	Date :

A faire vous-même II : Notre deuxième programme

Nous allons mettre en œuvre les quelques principes évoqués ci-avant.


Voici un programme dont il manque quelques instructions. En revanche, tous les commentaires sont fournis :

```

1  /* Libellé : programme_2.s                */
2  /* Objectif : retourne une valeur résultat*/
3  /* de l'addition de 2 entiers              */
4  /* En ARM, on utilise le mot clé "add"     */
5  /* pour additionner deux entiers          */
6  /* ex: add n1, n2, n3 ==> n1 ← n2 + n3    */
7
8  .global main          /* point d'entrée du programme */
9
10 main:                 /* programme principal */
11     |                  /* r4 ← 14 */
12     |                  /* r6 ← 17 */
13     |                  /* r0 ← r4 + r6 */
14     |                  /* fin du programme par retour à l'adresse contenue dans lr*/

```

- A l'aide de l'éditeur Geany, complétez le programme ci-dessus. Sauvegardez votre programme en utilisant le format <nom_programme>.s
- Assemblez le programme en utilisant le format <nom_programme>.o
- Editez les liens du programme
- Exécutez le programme <nom_programme>
- Quel est le résultat de la commande « echo \$? » ?

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Victor Hugo		
Nom :	Prénom :	Date :

A faire vous-même III : Données et mémoire

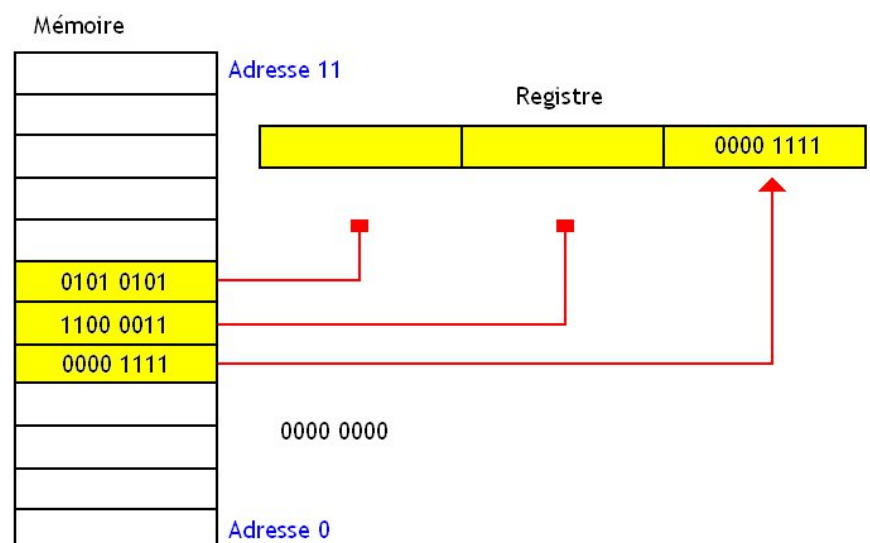
Nous avons vu précédemment que nous pouvions déplacer des valeurs dans des registres (en utilisant l'instruction « *mov* ») et ajouter deux registres (en utilisant l'instruction « *add* »). Si notre processeur ne pouvait travailler que sur des registres, nos actions seraient limitées.

Un ordinateur possède une mémoire dans laquelle sont stockés le programme et les données. En assembleur ARM, tous les opérandes doivent être des registres. Par conséquent, nous chargeons des données dans un registre à partir de la mémoire (load to register ou « *ldr* ») et nous stockons des données d'un registre dans une mémoire (store from register ou « *str* »).

Il existe plusieurs façons de charger et de stocker des données depuis / vers la mémoire, mais nous allons maintenant nous concentrer sur les plus simples: charger un registre « *ldr* » et stocker depuis un registre « *str* ».




Lorsqu'on évoque le chargement de données en mémoire, on évoque aussi l'adressage des données. L'adresse d'une donnée, c'est le nom de l'endroit précis où cette donnée est rangée dans la mémoire.

En assembleur ARM, l'adresse est un nombre de 32 bits qui identifie chaque octet de la mémoire.



Lors du chargement ou du stockage de données depuis/vers la mémoire, nous devons calculer une adresse. Cette adresse peut être calculée de plusieurs manières. Chacun de ces modes s'appelle un « mode d'adressage ». ARM possède plusieurs modes d'adressage. Dans cette activité, nous ne considérerons que l'adressage via un registre.

ARM possède des registres d'entiers de 32 bits et les adresses de la mémoire sont des nombres de 32 bits. Cela nous permet donc de conserver une adresse dans un registre. Une fois que nous avons une adresse dans un registre, nous pouvons utiliser ce registre pour charger ou stocker des données.

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS			
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur		
	Activité Pratique		
Nom : _____		Prénom : _____	Date : _____

Lorsque que nous avons expliqué le fonctionnement du premier programme, nous avons évoqué les étiquettes de l'assembleur. En fait, les étiquettes de l'assembleur ne sont que des noms symboliques d'adresses de votre programme. Ces adresses peuvent faire référence à la fois à des données ou à du code. Jusqu'ici, nous n'avons utilisé qu'un seul libellé « main » pour désigner l'adresse de notre « fonction » main. Une étiquette ne contient qu'une adresse, jamais son contenu.

Précédemment, nous avons écrit que l'assembleur était une couche mince au-dessus du code binaire. En fait cette couche n'est pas si mince que cela puisque l'outil assembleur (« as ») est responsable de l'affectation de valeurs aux adresses des étiquettes.

Ainsi, nous pouvons définir des données et attacher une étiquette à son adresse. En revanche, il nous appartient, en tant que programmeurs en assembleur, de nous assurer que le stockage référencé par l'étiquette possède la taille et la valeur appropriées.

Reprenons l'exemple de l'addition de deux valeurs en l'améliorant avec quelques accès à la mémoire :



Architecture matérielles
& Systèmes d'exploitation

A la découverte d'un Nano Ordinateur



Activité Pratique

Nom :



Prénom :

Date :

```

1  /* Libellé : programme_3.s */
2  /* Objectif : retourne une valeur résultat */
3  /* de l'addition de 2 entiers en accédant */
4  /* aux données stockées en mémoire */
5  /* En ARM, on utilise le mot clé "add" */
6  /* pour additionner deux entiers */
7  /* ex: add n1, n2, n3 ==> n1 ← n2 + n3 */
8
9
10 /****** */
11 /* Section Données */
12 .data
13
14 /* Le programmeur s'assure que la taille des données est de 4 octets ou 32 bits */
15 .balign 4
16
17 /* Attribution d'une étiquette à la variable entier1 */
18 entier1:
19
20 /* Initialisation de la variable entier1 avec la valeur '14' */
21 .word 14
22
23 /* On recommence avec entier2 */
24 .balign 4
25
26 /* Attribution d'une étiquette à la variable entier2 */
27 entier2:
28
29 /* Initialisation de la variable entier2 avec la valeur '17' */
30 .word 17
31
32 /****** */
33 /* Section code */
34 .text
35
36 /* Le programmeur s'assure que la taille du code est de 4 octets ou 32 bits */
37 .balign 4
38 .global main
39 main:
40     ldr r4, adr_de_entier1 /* r4 ← &entier1 */
41     ldr r4, [r4]          /* r4 ← *r4 */
42     ldr r6, adr_de_entier2 /* r6 ← &entier2 */
43     ldr r6, [r6]          /* r6 ← *r6 */
44     add r0, r4, r6        /* r0 ← r4 + r6 */
45     bx lr
46
47 /* étiquettes pour accéder aux données */
48 adr_de_entier1 : .word entier1
49 adr_de_entier2 : .word entier2

```

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : _____ Prénom : _____		Date : _____

Quelques explications...pour comprendre

Nous commençons par déclarer une variable de 4 octets à laquelle nous attribuons l'étiquette « entier1 ».

Ensuite, nous initialisons cette variable avec la valeur « 14 ».

Puis, nous déclarons une deuxième variable.

```

/*****/
/* Section Données */
.data

/* Le programmeur s'assure que la taille des données est de 4 octets ou 32 bits */
.balign 4

/* Attribution d'une étiquette à la variable entier1 */
entier1:

/* Initialisation de la variable entier1 avec la valeur '14' */
.word 14

/* On recommence avec entier2 */
.balign 4

/* Attribution d'une étiquette à la variable entier2 */
entier2:

/* Initialisation de la variable entier2 avec la valeur '17' */
.word 17

```

Il y a deux nouvelles directives d'assembleur dans l'exemple ci-dessus : « .balign » et « .word ». Une directive « .balign » s'assure que la prochaine adresse démarrera un espace mémoire de 4 octets car ARM impose certaines restrictions concernant les adresses des données que vous pouvez utiliser. C'est-à-dire que l'adresse du prochain point de référence émis (c'est-à-dire une instruction ou une donnée) sera un multiple de 4 octets. Cette directive ne fait rien si l'adresse était déjà alignée sur 4 octets. Sinon, l'outil d'assemblage émettra quelques octets de remplissage, non utilisés par le programme, de sorte que l'alignement demandé soit respecté. Cette directive peut être omise si toutes les entités émises par l'assembleur ont une largeur de 4 octets (soit 32 bits), mais dès que nous voulons utiliser des données de taille différente, cette directive devient obligatoire.

Ensuite Nous définissons maintenant l'adresse de « entier1 ». Grâce à l'ancienne directive « .balign », nous savons que son adresse sera alignée sur 4 octets.

Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS		
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : _____ Prénom : _____		Date : _____

Sections.

Les données sont stockées en mémoire comme du code, mais en pratique, elles sont généralement conservées ensemble dans ce que l'on appelle une section de données. « .data » indique à l'assembleur d'émettre les entités dans la section de données. La directive « .text » fait la même chose pour le code. Nous mettons donc les données après une directive « .data » et le code après un « .text ».

La directive « .word » indique que l'outil assembleur doit émettre la valeur de l'argument de la directive comme un entier de 4 octets.

Quelles seront les valeurs émises dans notre cas ? Pour quelles variables ?

.....
.....
.....
.....

Notez que nous comptons sur le fait que « .word » émet 4 octets pour définir la taille de nos données.

Chargement

Avant d'analyser les instructions de chargement des registres, nous allons nous arrêter sur deux nouvelles étiquettes.



Étiquettes de la section Code

```
/* étiquettes pour accéder aux données */
adr_de_entier1 : .word entier1
adr_de_entier2 : .word entier2
```

Les deux étiquettes « adr_de_entier1 » et « adr_de_entier2 », contiennent l'adresse de « entier1 » et « entier2 ». Mais pourquoi faire puisque nous avons déjà l'adresse de nos données dans les étiquettes de « entier1 » et « entier2 » ?

En fait, les étiquettes de « entier1 » et « entier2 » sont dans une section différente : dans la section « .data ». Cette section existe pour que le programme puisse la modifier, c'est pourquoi les variables y sont conservées. Par contre, le code n'est généralement pas modifié par le programme (pour des raisons d'efficacité et de sécurité). C'est donc une raison pour avoir deux sections différentes avec des propriétés différentes.

Mais nous ne pouvons pas accéder directement à un symbole d'une section à l'autre. Ainsi, nous avons besoin d'une étiquette spéciale qui se réfère à l'adresse d'une entité dans la section « .data ».

<div>Lycée d'enseignement général et technologique international Victor Hugo COLOMIERS</div> <div></div>		<div>NSI</div> <div>NUMÉRIQUE ET SCIENCES INFORMATIQUES</div>
Architecture matérielles & Systèmes d'exploitation	A la découverte d'un Nano Ordinateur	
	Activité Pratique	
Nom : Prénom : Date :		

Lorsque l'assembleur émet le code binaire, « .word entier1 » ne sera pas l'adresse de « entier1 » mais plutôt une adresse, dont la valeur exacte est inconnue, mais qui le sera lorsque le programme sera lié (c'est-à-dire lorsque l'exécutable final sera généré). Cela revient à dire que nous n'avons aucune idée de l'endroit où cette variable sera réellement stockée. Et nous laissons l'éditeur de liens « résoudre » cette valeur pour nous plus tard. L'adresse de « adr_de_entier1 » est dans la même section « .text ». Cette valeur sera « résolue » par l'éditeur de liens pendant la phase de liaison (lorsque l'exécutable final est créé et qu'il sait où toutes les entités de notre programme seront définitivement disposées en mémoire). C'est pourquoi l'éditeur de liens (invoqué en interne par gcc) est appelé ld, ce qui signifie « Link eDitor ».

Analyse des instructions de chargement des registres :

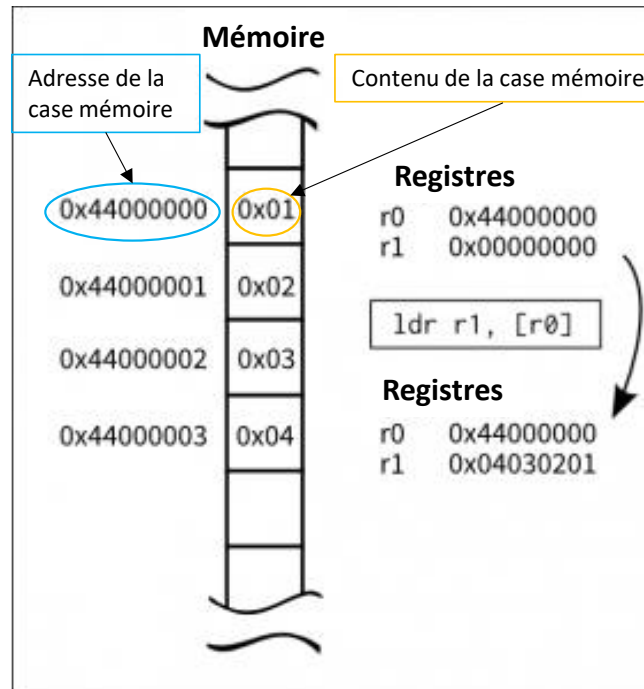
```

40      ldr r4, adr_de_entier1 /* r4 ← &entier1 */
41      ldr r4, [r4]          /* r4 ← *r4 */

ldr r1, addr_of_myvar1 /* r1 ← &myvar1 */
ldr r1, [r1]           /* r1 ← *r1 */

```

La ligne 40 charge la valeur de l'adresse de « entier1 » qui sera « résolue » par l'éditeur de liens. C'est-à-dire qu'il y a des données en mémoire, dont l'adresse est « adr_de_entier1 », avec une taille de 4 octets contenant l'adresse réelle de « entier1 ». Ainsi, après le premier « ldr », dans « r4 » nous avons l'adresse réelle de « entier1 ». Or, ce qui nous intéresse, c'est le contenu de la mémoire à cette adresse. Par conséquent, nous opérons un deuxième chargement ldr. Regardons ce qui se passe au niveau des registres en prenant un exemple ci-dessous :



Vous vous demandez probablement pourquoi les deux chargements ont une syntaxe différente. Le premier « ldr » utilise l'adresse symbolique du label « adr_de_entier1 ». Le second « ldr » utilise la valeur du registre comme mode d'adressage. Ainsi, dans le second cas, nous utilisons la valeur dans « r4 » comme adresse. Dans le premier cas, nous ne savons pas vraiment ce que l'assembleur utilise comme mode d'adressage, donc nous allons l'ignorer pour le moment.

Le programme charge deux valeurs 32 bits de « entier1 » et « entier2 », qui avaient les valeurs initiales 14 et 17, les ajoute et met le résultat de l'addition comme code d'erreur du programme dans le registre r0 juste avant de quitter main.

Selon vous, quelle sera la valeur du code erreur résultat ?

**** Fin du document ****