

Langage assembleur

Christophe Viroulaud

Première - NSI

Mach 04



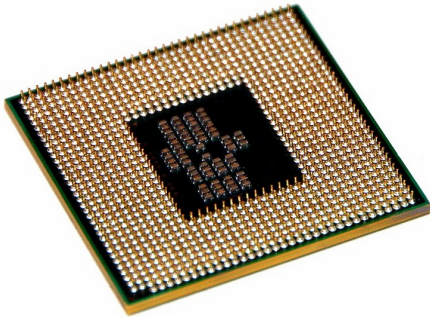


Figure 1 – Le processeur ne réagit qu'à des signaux électriques.

Communiquer avec le processeur.

1. Langage machine

1.1 Langage binaire

1.2 Retour historique

1.3 Langage de bas niveau

2. Langage assembleur

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

À retenir

- ▶ Un processeur est un composant électronique : il n'interprète que des signaux électriques.
- ▶ Il ne peut y avoir que deux états :
 - ▶ 1 : passage de courant électrique,
 - ▶ 0 : absence de courant électrique.

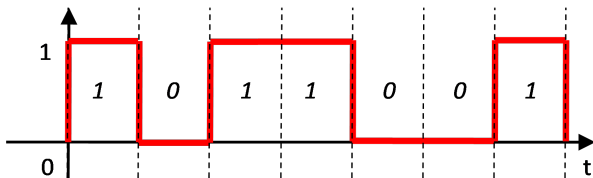


Figure 2 – On parle de **langage binaire**.

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

1. Langage machine

1.1 Langage binaire

1.2 Retour historique

1.3 Langage de bas niveau

2. Langage assembleur

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

Retour historique

Langage
assembleur

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation



1801

Un concept déjà existant : Joseph-Marie Jacquard a développé un métier à tisser avec lequel le motif à tisser était déterminé par des cartes perforées.

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

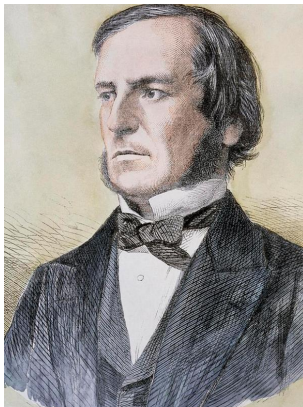
Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation



1844-1854

Une théorie mathématique : George Boole crée une algèbre binaire, dite booléenne, n'acceptant que deux valeurs numériques : 0 et 1.

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation



Figure 3 – 1936 : Alan Turing conceptualise une machine abstraite qui définit ce qui est réalisable par tout appareil mécanique de calculs.

1. Langage machine

1.1 Langage binaire

1.2 Retour historique

1.3 Langage de bas niveau

2. Langage assembleur

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

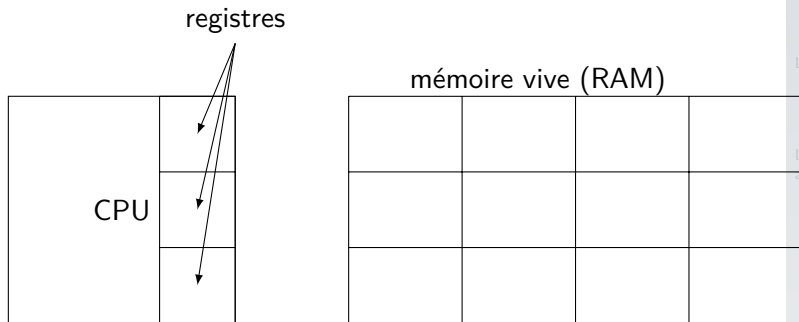
Opérations arithmétiques

Transfert de données

Documentation

À retenir

Au sens de Alan Turing, **un algorithme** est un ensemble d'étapes réalisables par la machine de Turing et qui permet de répondre à une problématique donnée.



À retenir

- ▶ Von Neumann s'est appuyé sur les concepts de Turing pour définir l'architecture d'un ordinateur.
- ▶ Des ajustements pratiques sont implémentés dans les machines réelles : le CPU ne peut manipuler que des valeurs stockées dans **les mémoires registres** : les fréquences de fonctionnement sont similaires.

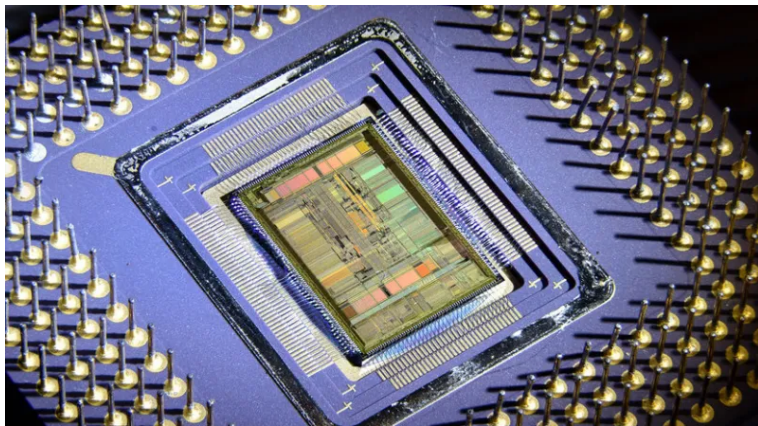


Figure 4 – Les registres sont très proches de l'UAL du processeur.

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

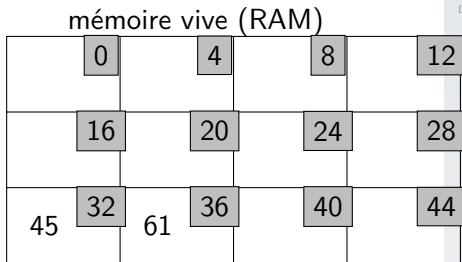
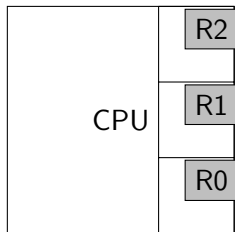
Transfert de données

Documentation

- ▶ Le processeur ne peut réaliser que des instructions basiques :
 - ▶ additionner deux valeurs,
 - ▶ comparer deux valeurs,
 - ▶ récupérer une valeur dans la mémoire vers le registre,
 - ▶ charger une valeur dans la mémoire depuis le registre,
 - ▶ lire une valeur entrée par l'utilisateur,
 - ▶ envoyer une valeur vers la sortie.

- ▶ Le processeur ne peut réaliser que des instructions basiques :
 - ▶ additionner deux valeurs,
 - ▶ comparer deux valeurs,
 - ▶ récupérer une valeur dans la mémoire vers le registre,
 - ▶ charger une valeur dans la mémoire depuis le registre,
 - ▶ lire une valeur entrée par l'utilisateur,
 - ▶ envoyer une valeur vers la sortie.
- ▶ Quand on se place à **bas-niveau**, écrire un algorithme consiste à détailler les étapes réalisables par le processeur.

Activité 1 : Donner la suite d'instructions à réaliser pour additionner les deux nombres stockées dans les emplacements 32 et 36 de la mémoire.



Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

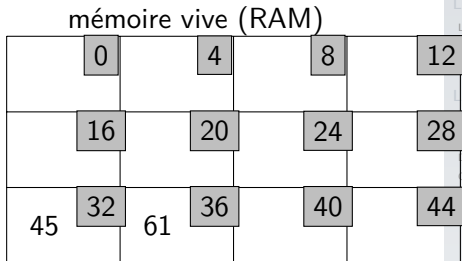
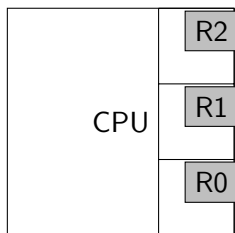
Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation



- ▶ Placer la valeur de la cellule 32 dans le registre R0.
- ▶ Placer la valeur de la cellule 36 dans le registre R1.
- ▶ Additionner R0 avec R1
- ▶ Placer le résultat dans R2.
- ▶ Placer le résultat dans la cellule 40.

Observation

Dans un langage machine, il faut 5 instructions pour écrire l'algorithme d'une addition.

1. Langage machine

2. Langage assembleur

2.1 Un langage intermédiaire

2.2 Découverte d'un simulateur

2.3 Opérations arithmétiques

2.4 Transfert de données

2.5 Documentation

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

Langage assembleur - Un langage intermédiaire

Langage
assembleur

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

1 11100010100000000001000000000101

Code 1 – Additionner le R0 à la valeur 5 et placer le résultat dans
R1

À retenir

Pour faciliter la communication avec le processeur, on remplace les codes binaires par des **symboles mnémoniques**.

ADD, MOV, SUB...

1. Langage machine

2. Langage assembleur

2.1 Un langage intermédiaire

2.2 Découverte d'un simulateur

2.3 Opérations arithmétiques

2.4 Transfert de données

2.5 Documentation

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

Activité 2 :

1. Ouvrir la page <https://edurl.fr/simul>
2. Repérer les éléments du modèle de von Neumann.

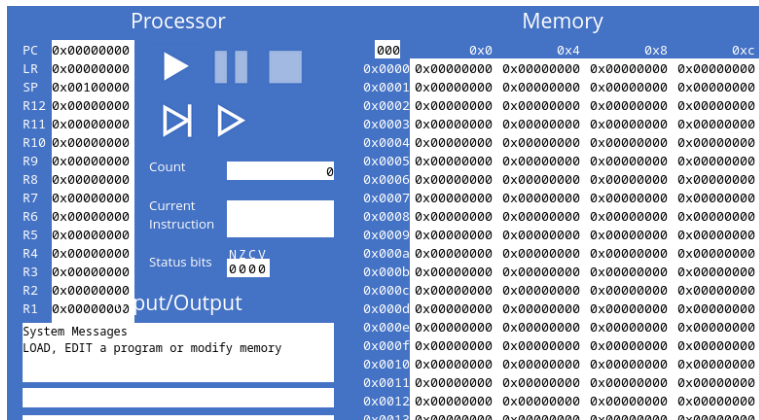


Figure 5 – Simulateur 32-bit ARM

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

Activité 3 :

1. Dans la partie *Program* cliquer sur **Edit** puis écrire les **instructions** suivantes :

```
1 MOV R0,#3
2 ADD R1,R0,#5
3 HALT
```

2. Cliquer sur **Submit** et observer ce qui se passe dans la mémoire.

Memory			
000	0x0	0x4	0x8
0x0000	0xe3a00003	0xe2801005	0xe1000070
0x0001	0x00000000	0x00000000	0x00000000
0x0002	0x00000000	0x00000000	0x00000000
0x0003	0x00000000	0x00000000	0x00000000
0x0004	0x00000000	0x00000000	0x00000000

Figure 6 – Chaque instruction est inscrite dans un **mot-mémoire**

Remarque

L'affichage hexadécimal est une représentation de l'instruction. On peut l'afficher en **langage binaire**.

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

Activité 4 : Cliquer sur le bouton pour exécuter le programme en mode **pas à pas**, afin de suivre son déroulement.



Figure 7 – Exécution pas à pas

The screenshot displays the Raspberry Pi 400 IDE interface, which is divided into two main sections: **Processor** and **Memory**.

Processor Section:

- Registers:** A list of registers (PC, LR, SP, R12, R11, R10, R9, R8, R7, R6, R5, R4, R3, R2, R1, R0) is shown on the left. The PC register contains the value 1048576.
- Control Panel:** A central area with icons for play, pause, stop, step over, step into, and settings.
- Count:** A progress bar labeled "Count" with the value 2.
- Current Instruction:** The instruction "ADD Rd,Rn,#im" is displayed.
- Status bits:** The status bits are shown as "NZCV" with the value "0000".

Memory Section:

- Memory View:** A table showing memory addresses and their corresponding values. The address 0x0000 is highlighted in orange, and the value -494923771 is highlighted in yellow.

Address	Value
0x0000	-476053501
0x0001	0
0x0002	0
0x0003	0
0x0004	0
0x0005	0
0x0006	0
0x0007	0
0x0008	0
0x0009	0
0x000a	0
0x000b	0
0x000c	0
0x000d	0
0x000e	0
0x000f	0

À retenir

En informatique, **une instruction** :

- ▶ s'écrit sur une ligne,
- ▶ modifie l'état de la mémoire (RAM, registre, bits de statut...)

1. Langage machine

2. Langage assembleur

2.1 Un langage intermédiaire

2.2 Découverte d'un simulateur

2.3 Opérations arithmétiques

2.4 Transfert de données

2.5 Documentation

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

```
1 ADD R0,R1,R2
```

Code 2 – Ajoute la valeur du registre R2 à celle de R1 puis place le résultat dans R0.

```
1 ADD R1,R1,#10
```

Code 3 – Ajoute l'entier 10 à celle du registre R1 puis place le résultat dans R1.

Remarque

Le symbole **SUB** effectue une soustraction avec la même syntaxe.

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

Activité 5 : Le compte est bon : Dans le jeu du *compte est bon* il faut retrouver un résultat à partir d'une série d'entiers. Dans cet exemple nous n'utiliserons que des additions et des soustractions. La série de nombre est : 12 20 57 3
Écrire un programme qui retrouve le résultat 52.

```
1 ADD R0,R0,#57
2 ADD R0,R0,#3
3 ADD R0,R0,#12
4 SUB R0,R0,#20
5 HALT
```

Code 4 – Un code possible

Remarque

On suppose ici que le registre contient la valeur 0 au démarrage du programme.

1. Langage machine

2. Langage assembleur

2.1 Un langage intermédiaire

2.2 Découverte d'un simulateur

2.3 Opérations arithmétiques

2.4 Transfert de données

2.5 Documentation

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

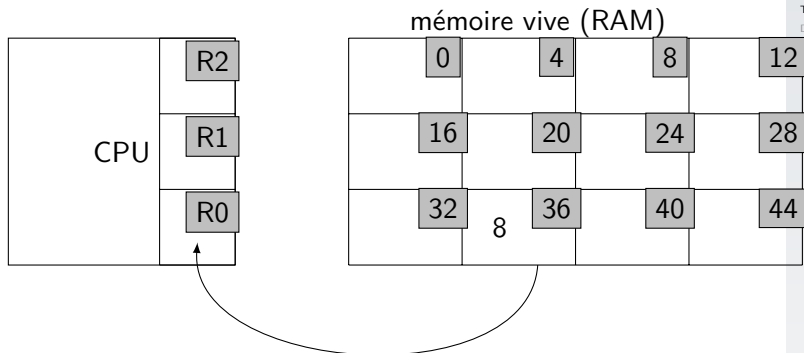
Transfert de données

Documentation

Transfert de données

1 LDR R0,36

Code 5 – **LoaDR**egister : charge dans R0 la valeur située à l'adresse 36 de la mémoire.



Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

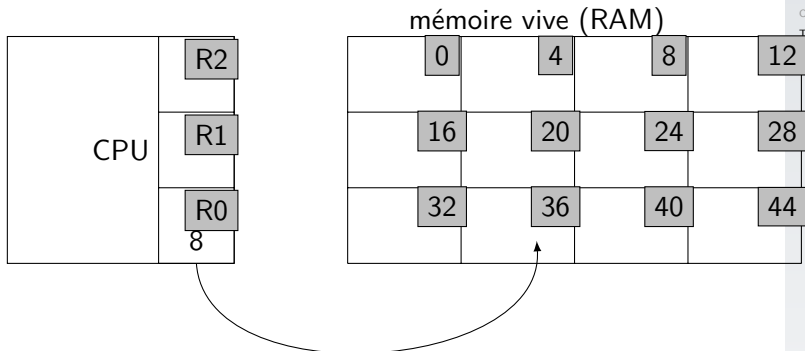
Opérations arithmétiques

Transfert de données

Documentation

1 STR R0,36

Code 6 – **SToreRegister** : stocke la valeur de R0 dans l'espace mémoire situé à l'**adresse** 36.



Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

Activité 6 : Compléter le programme précédent pour qu'il stocke le résultat final (52) dans la case mémoire 32.

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

```
1  ADD R0,R0,#57
2  ADD R0,R0,#3
3  ADD R0,R0,#12
4  SUB R0,R0,#20
5  STR R0,32
6  HALT
```

1. Langage machine

2. Langage assembleur

2.1 Un langage intermédiaire

2.2 Découverte d'un simulateur

2.3 Opérations arithmétiques

2.4 Transfert de données

2.5 Documentation

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

À retenir

Chaque processeur possède son propre jeu d'instructions. Il n'est pas possible de les connaître par cœur. La documentation est donc un outil indispensable pour le programmeur.

Langage machine

Langage binaire

Retour historique

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Documentation

Activité 7 :

1. Cliquer sur le bouton **Documentation** en bas à droite du simulateur.
2. Ouvrir le lien **ARMLite Programming Reference Manual**.