

Programmation assembleur

Christophe Viroulaud

Première - NSI

ArchMat 02

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

L'*unité de contrôle* d'un processeur lit des instructions contenues dans la mémoire et ordonne à l'*unité arithmétique et logique* de les exécuter. Cependant, le processeur ne comprend pas le langage humain et des consignes même simples ne sont pas directement interprétables par la machine.

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Comment l'Homme communique avec la machine ?

1. Langage machine

1.1 Langage binaire

1.2 Langage de bas niveau

2. Langage assembleur

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

À retenir

Un processeur est un composant électronique : il n'interprète que des signaux électriques (représentés par 0 ou 1).

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

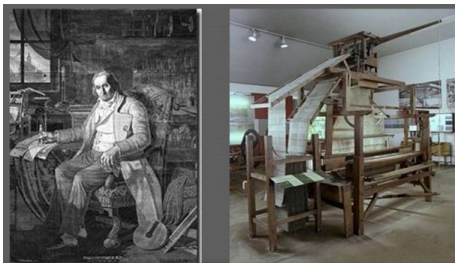
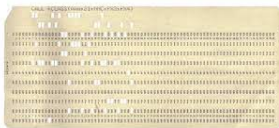
Entrées / Sorties

Techniquement il ne peut y avoir que deux états :

- ▶ passage de courant électrique représenté par le chiffre 1,
- ▶ absence de courant électrique représenté par le chiffre 0.

On parle de **langage binaire**.

Un concept déjà existant



1801

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

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

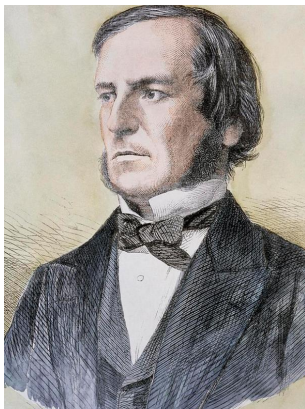
Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Une théorie mathématique



1844-1854

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

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

1. Langage machine

1.1 Langage binaire

1.2 Langage de bas niveau

2. Langage assembleur

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

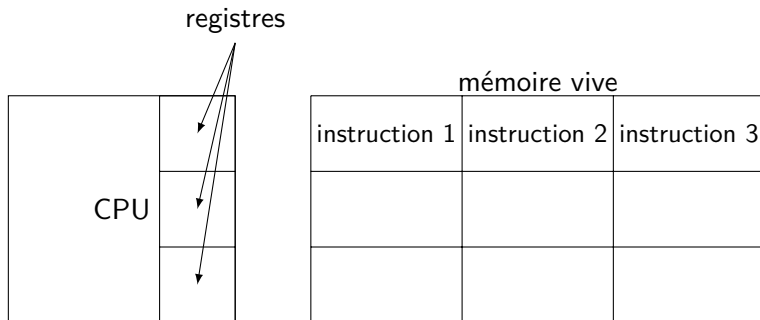


FIGURE 1 – Rappel : modèle de von Neumann

Un processeur ne peut exécuter que des instructions
basiques :

- ▶ **opérations arithmétiques** : « *additionne la valeur contenue dans le registre R1 et le nombre 789 et range le résultat dans le registre R0* »

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Un processeur ne peut exécuter que des instructions basiques :

- ▶ **opérations arithmétiques** : « *additionne la valeur contenue dans le registre R1 et le nombre 789 et range le résultat dans le registre R0* »
- ▶ **transfert de données entre les registres et la mémoire vive** : « *prendre la valeur située à l'adresse mémoire 487 et la placer dans la registre R2* »

Un processeur ne peut exécuter que des instructions basiques :

- ▶ **opérations arithmétiques** : « *additionne la valeur contenue dans le registre R1 et le nombre 789 et range le résultat dans le registre R0* »
- ▶ **transfert de données entre les registres et la mémoire vive** : « *prendre la valeur située à l'adresse mémoire 487 et la placer dans la registre R2* »
- ▶ **rupture de séquence** : « *saute de l'instruction 2 à l'instruction 5* »

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 Rupture de séquence

2.6 Entrées / Sorties

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Le code *0010 0110* donne l'ordre au processeur d'effectuer une multiplication.

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

À retenir

Pour faciliter la vie des informaticiens, on remplace les code 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 Rupture de séquence

2.6 Entrées / Sorties

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Activité 1 :

1. Ouvrir la page <https://www.peterhigginson.co.uk/ARMLite/>
2. Repérer les éléments du modèle de von Neumann.

Correction

The screenshot displays a 32-bit ARM simulator interface. It is divided into three main sections: Processor, Memory, and Input/Output.

Processor Section: Lists registers PC through R0, each with a value of 0x00000000. It includes control buttons for play, pause, and step-through. A 'Count' field is set to 0. The 'Current Instruction' field is empty. The 'Status bits' are shown as NZCV 0000.

Memory Section: A table of memory addresses from 0x0000 to 0x001f. The first address (0x0000) contains the value 000. All other addresses contain 0x00000000. The columns are labeled 0x0, 0x4, 0x8, and 0xc.

Input/Output Section: A text area containing 'System Cleared'.

At the bottom, there is a 'Hex' dropdown menu and a 'Clear' button.

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

FIGURE 2 – Simulateur 32-bit ARM

Activité 2 :

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 en mémoire.
3. Cliquer sur le bouton **3** pour exécuter le programme.



FIGURE 3 – Exécution pas à pas

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

- ▶ Le processeur exécute les instructions les unes après les autres.
- ▶ Les **mots-mémoires** sont présentés en *hexadécimal*.

Commentaire

Dans cette première approche, nous afficherons les mots-mémoires et les données en

Decimal (signed)

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 Rupture de séquence

2.6 Entrées / Sorties

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

À retenir

Un processeur ne peut effectuer des calculs qu'avec des valeurs situées dans ses registres.

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

Code 1 – Ajoute la valeur du registre R2 à celle de R1 puis place le résultat dans R0.

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

Code 2 – Ajoute l'entier 10 à celle du registre R1 puis place le résultat dans R1.

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

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Activité 3 : 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.

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

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

Code 3 – Un code possible

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 Rupture de séquence

2.6 Entrées / Sorties

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Il est possible d'envoyer une valeur directement dans un registre.

```
1 MOV R0, #10
```

Code 4 – Place la valeur 10 dans R0.

Il arrive régulièrement que les données soient déjà stockées dans la mémoire vive. Avant d'effectuer une opération arithmétique avec ces valeurs, il faut d'abord les charger dans les registres.

```
1 LDR R0,16
```

Code 5 – **LoaDR**egister : charge dans R0 la valeur située à l'adresse 16 de la mémoire.

```
1 STR R0,20
```

Code 6 – **ST**ore**R**egister : stocke la valeur de R0 dans l'espace mémoire situé à l'**adresse** 20.

Il semble fastidieux de demander au programmeur de manipuler les adresses mémoires. Heureusement il est possible d'assigner un **label** à un mot-mémoire.

```

1 LDR R0, mavaleur
2 HALT
3 mavaleur: 10
    
```

Code 7 – charge dans R0 la valeur située dans la case mémoire *mavaleur*.

Activité 4 :

1. Tester le programme précédent. Observer la valeur stockée en mémoire.
2. Écrire un programme qui :
 - ▶ stocke dans la mémoire vive les coordonnées $x = 5$ et $y = 4$ d'un personnage dans un jeu.
 - ▶ modifie chaque coordonnée de 3 unités.

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

```
1 LDR R0, x
2 ADD R0, R0, #3
3 STR R0, x
4 LDR R1, y
5 ADD R1, R1, #3
6 STR R1, y
7 HALT
8 x: 5
9 y: 4
```

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

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 Rupture de séquence

2.6 Entrées / Sorties

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Rupture de séquence

Les codes construits précédemment sont exécutés linéairement. Il peut être nécessaire de *sauter* à certains points de code.

```

1  MOV R0, #10
2  MOV R1, #10
3  // Compare les valeurs de R0 et R1
4  CMP R0, R1
5  // Si les valeurs sont égales, saute au label
6  BEQ labelegal
7  MOV R2, R0
8  HALT
9  labelegal:
10 STR R0, mavaleur
11 HALT
12 mavaleur: 5
    
```

Code 8 – Les // permettent de commenter le code.

Langage machine

Langage binaire

Langage de bas niveau

 Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Activité 5 :

1. Tester le code précédent en mode pas à pas.
Prendre le temps de bien comprendre le saut.
2. Dans le code remplacer la valeur dans R1 par 11.
Observer alors l'exécution du code.
3. Le **HALT** en ligne 8 est-il utile ?
4. Cliquer sur *Documentation* en bas à droite de l'écran. Sur la nouvelle page cliquer sur le lien *ARMLite Programming Reference Manual*.
5. Dans le manuel de référence, trouver les différents sauts (branchements) possibles.

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

- ▶ L'instruction **BEQ** en ligne 6 effectue un saut jusqu'à la ligne 9. Le code des lignes 7 et 8 n'est pas exécuté.
- ▶ Si les valeurs de R0 et R1 ne sont pas égales, **BEQ** ne fait rien et la ligne 7 sera la prochaine exécutée.
- ▶ Le **HALT** en ligne 8 est indispensable, sinon les lignes 9, 10, 11 seront exécutées.
- ▶ Dans la documentation en pages 7-8 on trouve :
 - ▶ **B** : unconditional branch,
 - ▶ **BNE** : Branch if Not Equal,
 - ▶ **BLT** : Branch if Less Than,
 - ▶ **BGT** : Branch if Greater Than.

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

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 Rupture de séquence

2.6 Entrées / Sorties

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Dans le modèle de von Neumann, pour communiquer avec l'utilisateur l'ordinateur utilise des interfaces d'entrée et de sortie.

```

1 // charge une entrée clavier (un nombre) dans R0
2 LDR R0, .InputNum
    
```

Code 9 – Le programme est en attente d'une entrée dans la console.

```

1 // Copie du texte GAGNE dans R2
2 MOV R2,#GAGNE
3 // affiche dans la console de sortie
4 STR R2, .WriteString
5 HALT
6 GAGNE:.ASCIZ "Bien joué!"
    
```

Code 10 – Le programme affiche le message dans la console de sortie.

Langage machine

Langage binaire

Langage de bas niveau

Langage
assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

Activité 6 : Écrire un programme qui :

- ▶ demande à l'utilisateur un nombre entre 1 et 10,
- ▶ le compare à un nombre préalablement chargé en mémoire, par le programmeur,
- ▶ affiche *Bravo* si l'utilisateur trouve le bon nombre,
- ▶ affiche *Perdu* sinon.

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties

```
1  LDR R0,.InputNum // Copie d'une entrée clavier dans
   R0
2  MOV R1,#10 // Charge 10 dans R1
3  CMP R0,R1 // Compare RO et R1
4  BNE NOTEGAL // S'il n'y a pas égalité, aller au
   label NOTEGAL
5  MOV R2,#GAGNE // Copie du texte GAGNE dans
   R2
6  STR R2,.WriteString // Affiche R2 dans la sortie
7  B SUITE // Aller au label SUITE
8  NOTEGAL:
9  MOV R2,#PERDU
10 STR R2,.WriteString
11 SUITE:
12 HALT
13 GAGNE:.ASCIZ "Bien joué!"
14 PERDU: .ASCIZ "Perdu!"
```

- ▶ <http://www.lunil.com/technologie-cartes-perforees-informatiques/>
- ▶ <https://info.blaisepascal.fr/>

Langage machine

Langage binaire

Langage de bas niveau

Langage assembleur

Un langage intermédiaire

Découverte d'un simulateur

Opérations arithmétiques

Transfert de données

Rupture de séquence

Entrées / Sorties