

TP Chapitre 06 : Pratique de l'Assembleur

Objectifs pratiques

- Écrire des programmes en assembleur A32
- Utiliser le simulateur web
- Déboguer avec le visualiseur
- Comprendre l'encodage binaire

Durée estimée : 2h

Prérequis : TD Chapitre 06 terminé

Partie 1 : Prise en Main du Simulateur

Accès au Simulateur

Ouvrir le Simulateur HDL

Interface :

- Éditeur de code (gauche)
- Registres et mémoire (droite)
- Boutons : Run, Step, Reset

Exercice 1 : Premier Programme

Objectif : Exécuter un programme simple

Code

```
; Premier programme : addition  
MOV R0, #5      ; R0 = 5  
MOV R1, #3      ; R1 = 3  
ADD R2, R0, R1  ; R2 = R0 + R1  
HALT
```

Étapes

1. Copiez le code dans l'éditeur
2. Cliquez sur **Reset**
3. Cliquez sur **Step** pour exécuter instruction par instruction
4. Observez R0, R1, R2 changer

Validation

R2 = 8

Exercice 2 : Boucle Simple

Objectif : Écrire une boucle comptant de 0 à 5

Code à compléter

```
MOV R0, #0      ; Compteur = 0
loop:
; TODO: Incrémenter R0
; TODO: Comparer R0 à 5
; TODO: Boucler si R0 <= 5
HALT
```

Indice

Utilisez : ADD , CMP , B.LE

► Solution

Exercice 3 : Somme des Entiers

Objectif : Calculer la somme $1 + 2 + \dots + N$

Spécification

- N est dans R0 (ex: N = 10)
- Résultat dans R1

Code à compléter

```
MOV R0, #10      ; N = 10
MOV R1, #0       ; Somme = 0
MOV R2, #1       ; i = 1
loop:
    ; TODO: Ajouter i à la somme
    ; TODO: Incrémenter i
    ; TODO: Comparer i à N
    ; TODO: Boucler si i <= N
HALT
```

► Solution

Exercice 4 : Multiplication

Objectif : Multiplier deux nombres par additions successives

Spécification

- R0 = premier nombre (ex: 7)
- R1 = deuxième nombre (ex: 6)
- R2 = résultat

Algorithme

```
resultat = 0
tant que R1 > 0:
    resultat += R0
    R1--
```

Code à écrire

Écrivez le programme complet.

► Solution

Partie 2 : Mémoire et Tableaux

Accès Mémoire

Instruction	Effet
LDR Rd, [Rn]	$Rd = \text{MEM}[Rn]$
STR Rd, [Rn]	$\text{MEM}[Rn] = Rd$
LDR Rd, [Rn, #off]	$Rd = \text{MEM}[Rn + off]$

Exercice 5 : Lecture/Écriture Mémoire

Objectif : Manipuler la mémoire

Code

```
.text
    LDR R0, =data    ; Adresse de data
    LDR R1, [R0]      ; Charger valeur
    ADD R1, R1, #1    ; Incrémenter
    STR R1, [R0]      ; Sauvegarder
    HALT

.data
data:   .word 42
```

Questions

1. Que vaut R1 après le LDR ?
2. Que contient la mémoire à l'adresse **data** à la fin ?

► Réponses

Exercice 6 : Parcours de Tableau

Objectif : Trouver le maximum d'un tableau

Code à compléter

```
.text
    LDR R0, =array ; Adresse tableau
    MOV R1, #5      ; Taille
    MOV R2, #0      ; Max actuel
loop:
    LDR R3, [R0]    ; Charger élément
    ; TODO: Si R3 > R2, mettre à jour R2
    ADD R0, R0, #4  ; Élément suivant
    SUB R1, R1, #1  ; Décrémenter compteur
    CMP R1, #0
    B.NE loop
    HALT

.data
array: .word 3, 7, 2, 9, 4
```

► Solution

Partie 3 : Fonctions

Convention d'appel simplifiée

- Arguments : R0, R1, R2, R3
- Retour : R0
- Sauvegarde : LR (Link Register = R14)
- Appel : `BL fonction`
- Retour : `BX LR`

Exercice 7 : Fonction Simple

Objectif : Écrire une fonction qui double un nombre

Code

```
.text
    MOV R0, #21      ; Argument
    BL double        ; Appeler la fonction
    HALT            ; R0 contient le résultat

; Fonction double(x) : retourne x * 2
double:
    ADD R0, R0, R0  ; R0 = R0 + R0
    BX LR           ; Retour
```

Exécutez et vérifiez

R0 final = 42

Exercice 8 : Factorielle (Défi)

Objectif : Calculer $n!$ de manière itérative

Spécification

- Entrée : $R_0 = n$
- Sortie : $R_0 = n!$

Algorithmme

```
result = 1
while n > 1:
    result = result * n
    n = n - 1
return result
```

► Solution

Partie 4 : Débogage

Techniques de Débogage

- 1. Step** : Exécuter une instruction à la fois
- 2. Breakpoint** : S'arrêter à une ligne précise
- 3. Watch** : Observer un registre ou une adresse
- 4. Print** : Écrire dans la sortie (si disponible)

Exercice 9 : Trouver le Bug

Objectif : Corriger un programme bugué

Code (avec bug)

```
; Calcule la somme de 1 à 5
MOV R0, #0      ; Somme
MOV R1, #1      ; i
loop:
    ADD R0, R0, R1
    ADD R1, R1, #1
    CMP R1, #5
    B.LT loop     ; BUG ICI !
    HALT
```

Quel est le bug ? Quelle est la valeur incorrecte de R0 ?

► Solution

Exercice 10 : Programme Libre

Objectif : Écrire un programme de votre choix

Suggestions

1. **Fibonacci** : Calculer le n-ième nombre de Fibonacci
2. **Palindrome** : Vérifier si un tableau est un palindrome
3. **Tri** : Trier un petit tableau (bubble sort)
4. **PGCD** : Calculer le PGCD de deux nombres (Euclide)

Critères

- Utilise au moins une boucle
- Utilise au moins un accès mémoire
- Comporte des commentaires

Récapitulatif

Compétences validées

- [] Écrire un programme assembleur simple
- [] Utiliser les boucles et conditions
- [] Accéder à la mémoire (tableaux)
- [] Écrire et appeler des fonctions
- [] Déboguer un programme

Prochaine étape

Chapitre 07 : Le Compilateur — Du C32 vers l'assembleur !

Référence : Livre Seed, Chapitre 06 - Assembleur