**Lab #4 : Organisation d’un ordinateur de base**

**CEG 2536 – ARCHITECTURE DES ORDINATEURS I**

**Fall 2018**

**Ecole de Génie Electrique et Science Informatique**

**Université d’Ottawa**

Professeur : Dr. Mohamed Ali Ibrahim

Groupe

Etudiant 1 : Rayold Rakotonomenjanahary 8884585

Etudiant 2 : Mahmoud Lafdaoui 300032370

# PARTIE THEORIQUE

## Introduction

Après avoir parlé de long en large de l’architecture de l’ordinateur de Mano, il était temps d’apprendre à le programmer en langage Assembleur. Le but de ce laboratoire était donc de permettre aux étudiants de très bien comprendre comment est conçu l’ordinateur de base; ensuite, il fallait concevoir, compiler, simuler le circuit contenant l’unité de contrôle et écrire un programme en langage d’assemblage, le convertir en code machine puis le simuler dans Quartus et sur la carte Altera.

Comme d’habitude nous avons eu droit à un certain nombre de matériel tel que :

* Quartus II (édition de l'étudiant ou l'édition web)
* la Carte Altera DE2-115 avec
* Câble USB-Blaster
* Alimentation 12 VDC, 2A.

C’est une occasion pour nous de mettre au point le contrôleur d’un ordinateur de base, et utiliser des opcodes pour y écrire des programmes simples.

## Discussion de solutions algorithmiques

Pour pouvoir concevoir l’unité de commande il nous fallait trouver les conditions sous lesquelles on chargeait dans tous les registres de l’unité de commande.

Voici les équations obtenues pour l’unité de commande :

Par la suite nous devrions écrire un programme en code machine. Le but de ce programme était d’additionner une série de nombres placés dans la mémoire et s’arrêter une fois que la somme devient égale à zéro et retourner le dernier nombre ajouté.

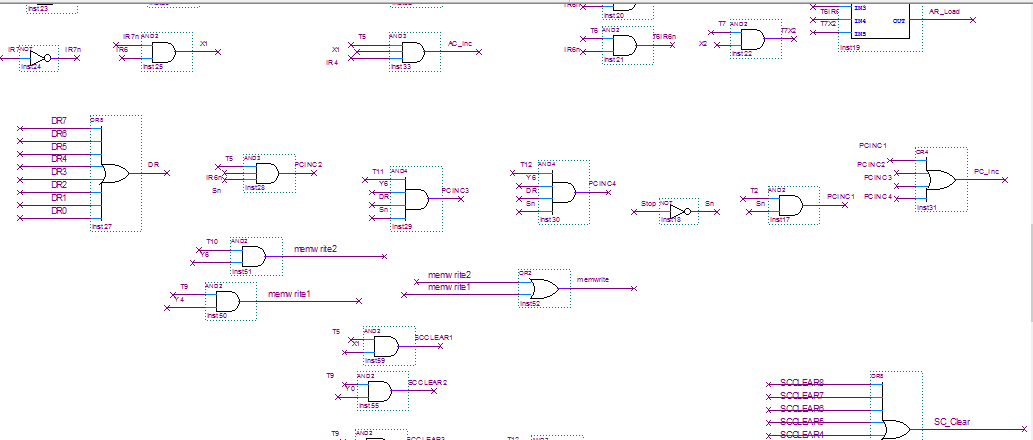
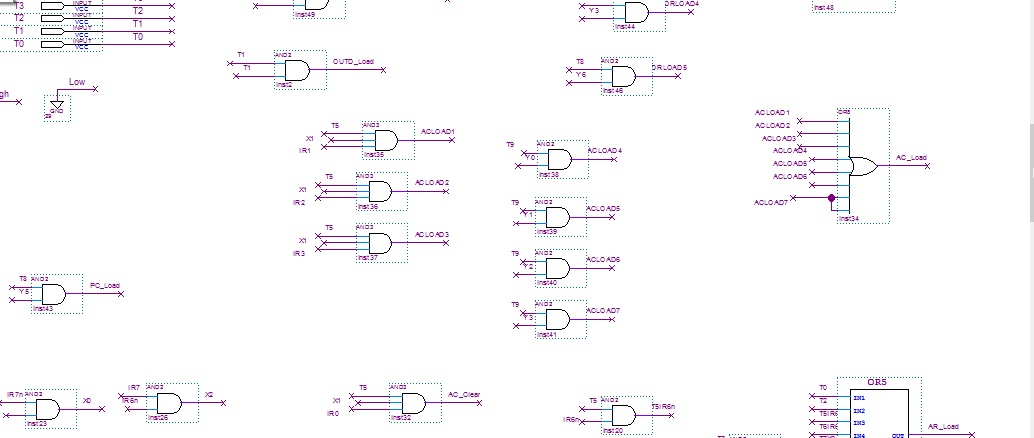
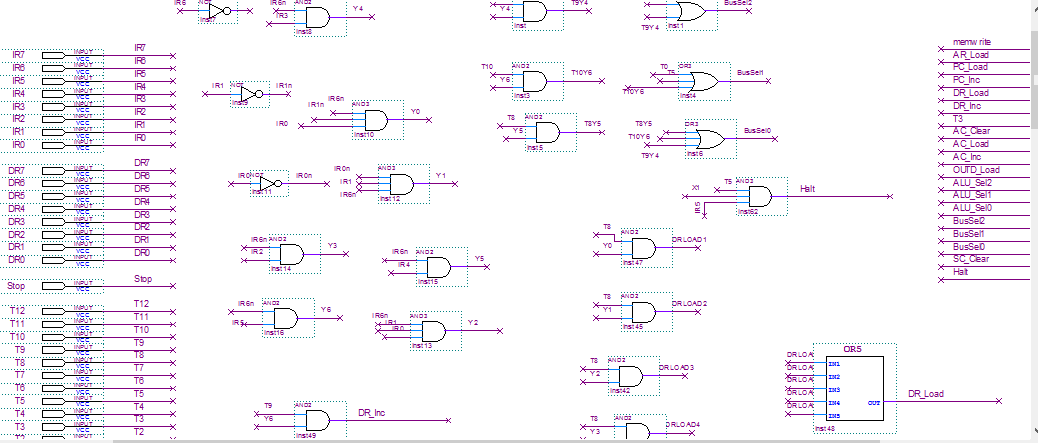
Pour ce nous avons d’abord écrit un pseudo code pour trouver une logique à appliquer ensuite converti ce pseudo code en langage assembleur en utilisant que les instructions supportées par l’ordinateur de base et enfin convertir ce code en code machine.

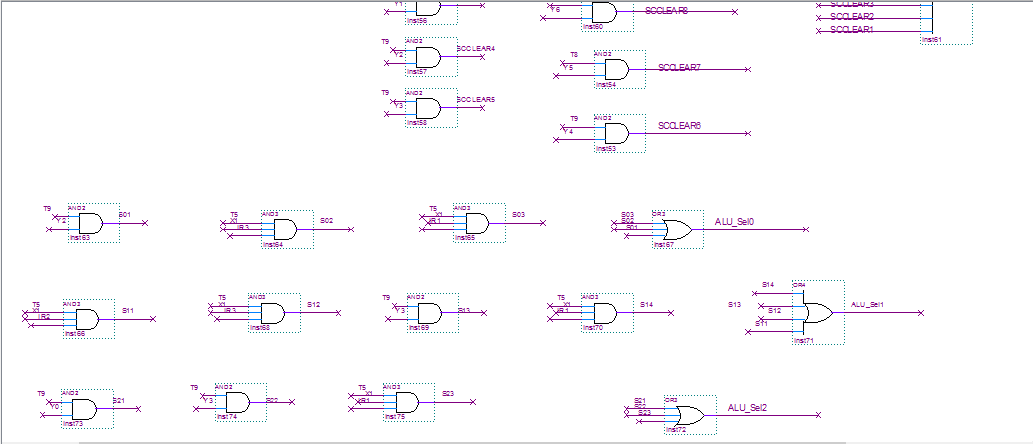
## Discussion de problèmes

Le plus grand problème rencontré lors de cette analyse était le grand nombre d’équations à trouver et les erreurs que nous avons commises qui étaient difficiles à repérer lorsque la simulation ne fonctionnait pas comme prévu.

En utilisant ces équations nous avions pu concevoir notre unité de commande.

Dans la partie conception, la difficulté rencontrée était qu’il fallait se servir seulement des instructions de cet ordinateur de base.



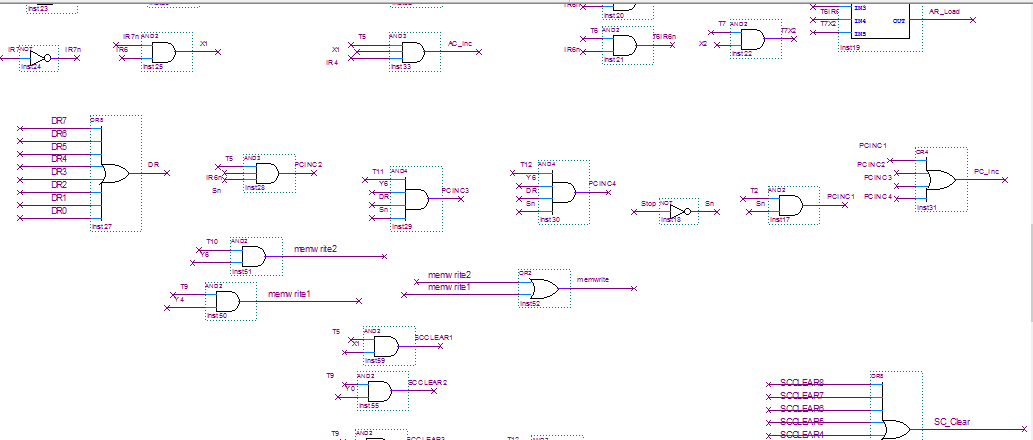
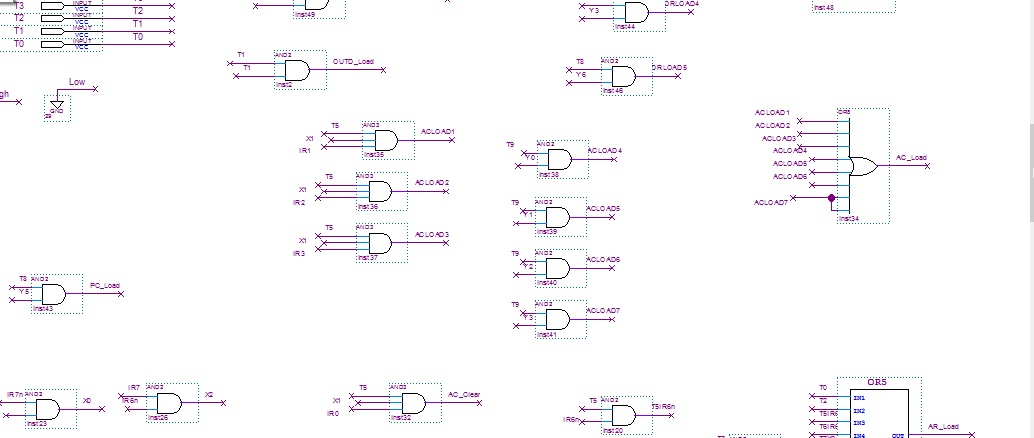
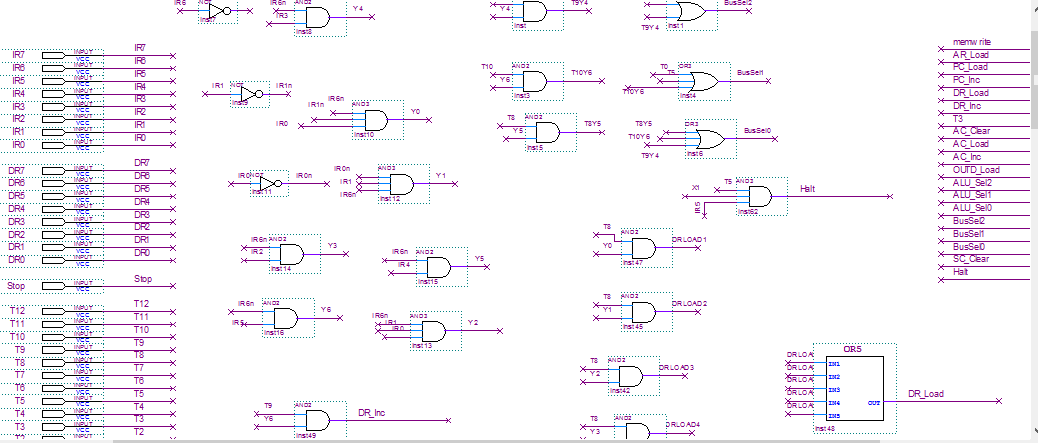


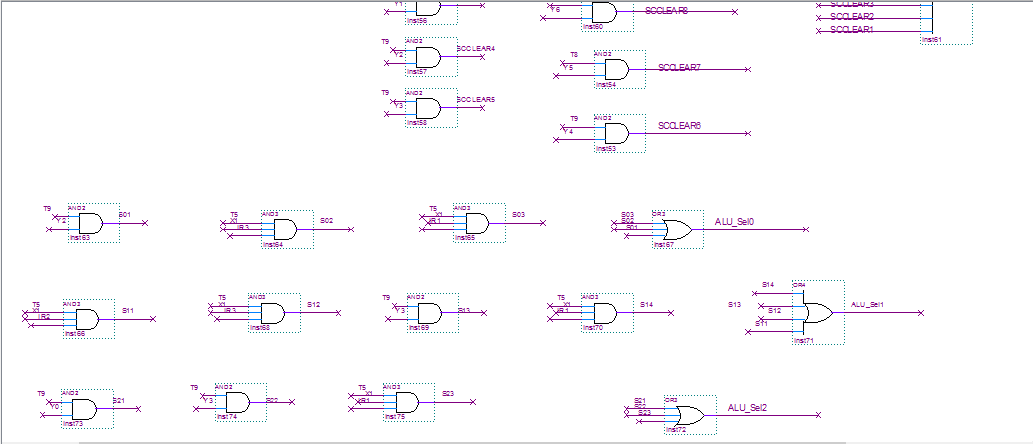
# **PARTIE CONCEPTION**

L’unité de commande est une partie clé dans la conception d’un ordinateur car sans lui il n’y aurait aucune logique et aucun ordre dans le processeur : tous les registres écriraient sur le bus à n’importe quel moment et dans tous les autres registres. C’est pourquoi concevoir une unité de commande parfaite est indispensable pour avoir un ordinateur qui fonctionne correctement.

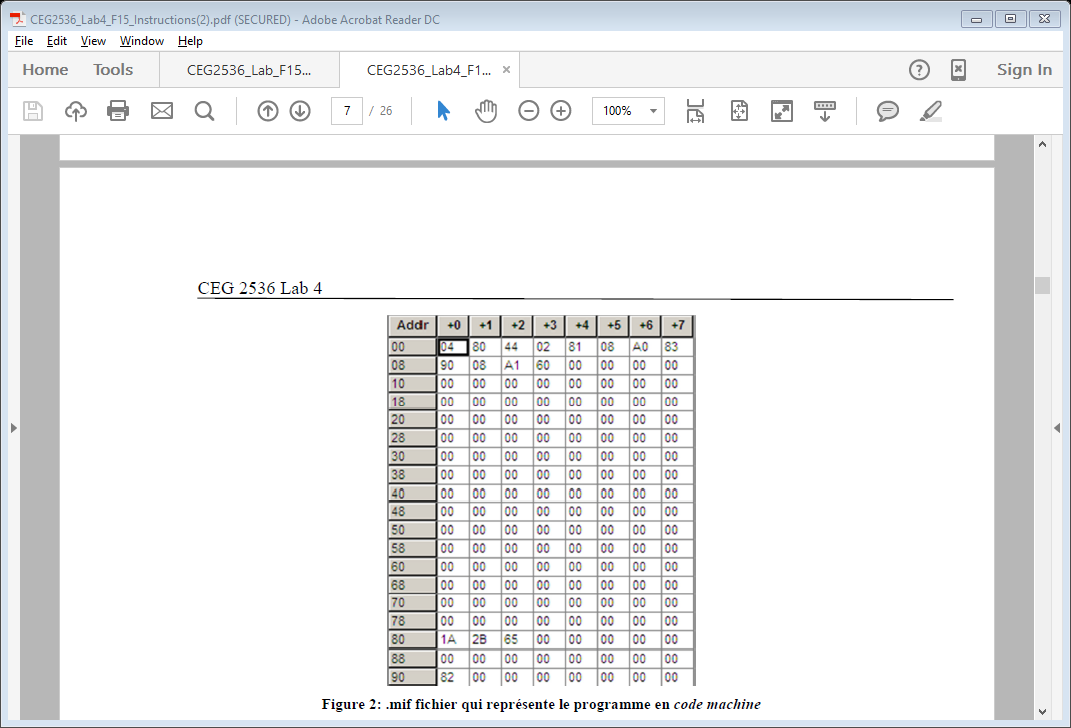
Composants utilisés

Conception de l’unité de contrôle  
Pour concevoir notre unité de commande, nous nous sommes servis des équations trouvées plus haut et ces équations ont été implémentées grâce à des portes Ou, Et et Non.





Nous devions aussi rajouter une sortie pour le bus OUTD dans le fichier global de l’ordinateur de base pour savoir quelle information passait par ce bus durant la simulation et aussi pour la programmation. Puis, on a créé un fichier de mémoire de taille 256x8 et nous y avons inscrit un code qui était donné dans les instructions du laboratoire. (Voir figure ci-dessous)



Un problème rencontré pour cette partie de l’expérience c’est que le fichier .mif contenait des fautes donc il fallait tout refaire le fichier .mif pour que la simulation le prenne bien en compte pour montrer les résultats.

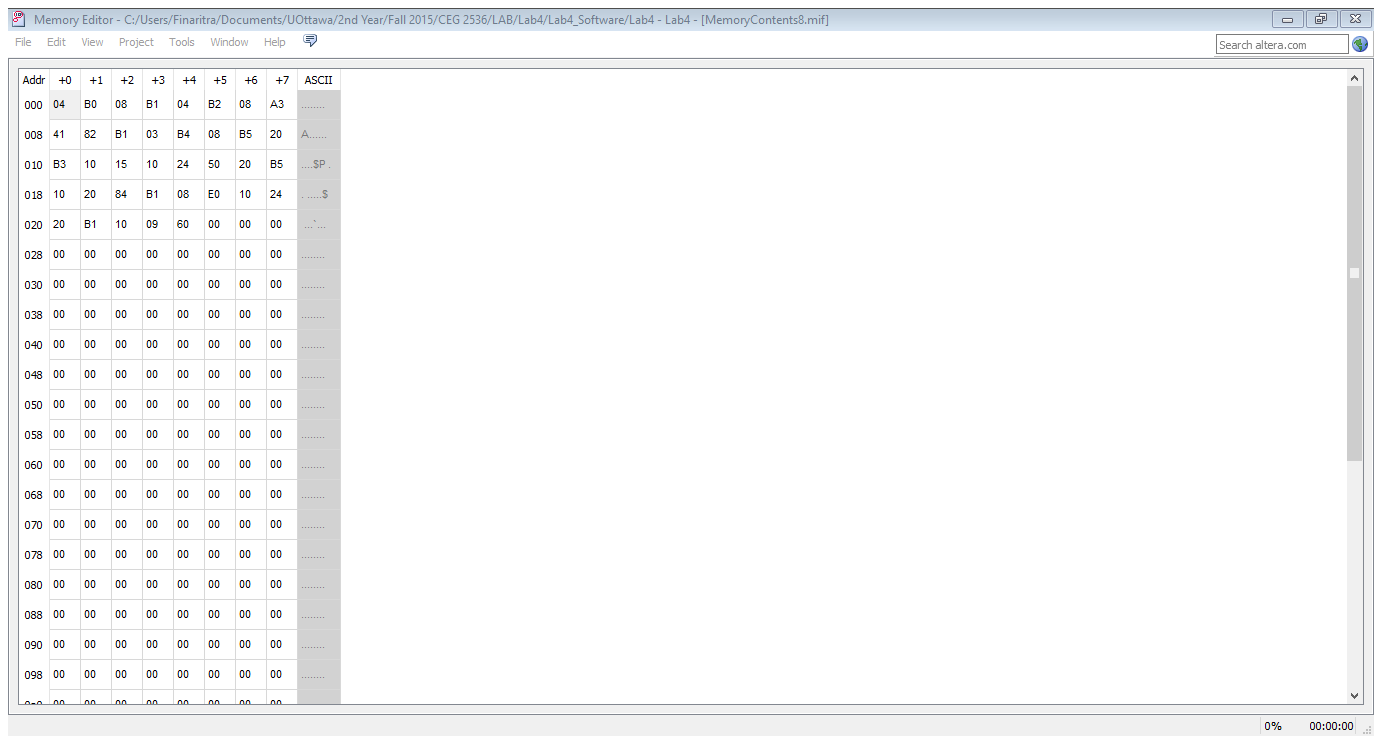
Conception du programme:  
Pour la conception du programme faisant la somme des nombres jusqu’à atteindre 0, nous avons d’abord utilisé le pseudocode suivant écrit en assemblé.

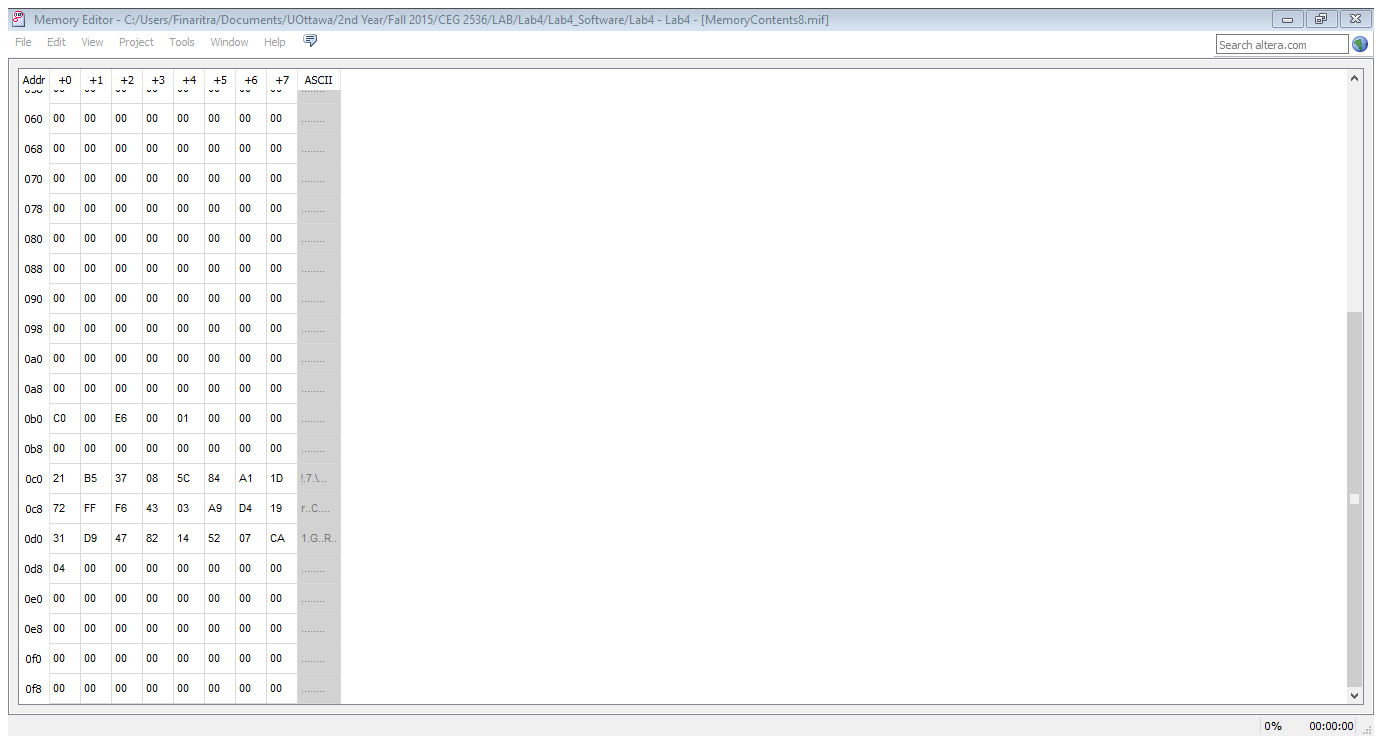
Le code en assemblé :

|  |  |  |
| --- | --- | --- |
|  | LDA | VLP |
|  | STA | PTR |
|  | LDA | VLC |
|  | STA | CTR |
|  | CLA |  |
| LOP | ADD | PTR I |
|  | SUB | VLS |
|  | STA | SUM |
|  | ISZ | CTR |
|  | BUN | CTN |
|  | BUN | STP |
| CTN | INC |  |
|  | ISZ | SUM |
|  | BUN | CT1 |
|  | LDA | PTR I |
|  | STA | E0 |
|  | BUN | STP |
|  | ISZ | PTR |
|  | BUN | LOP |
|  | HLT |  |
|  |  |  |
| A0 VLP | HEX 0080 |  |
| A1 PTR | 0000 |  |
| A2 VLC | DEC -26 |  |
| A3 CTR | 0000 |  |
| A4 VLS | DEC -1 |  |
| A5 SUM | 0000 |  |
| E0 | 0000 |  |
| C0 à D8 | Valeurs à additionner | |

Un problème que nous avions rencontré pendant la conception de ce programme était que l’ordinateur   
ne possédait pas l’instruction de vérifier si l’accumulateur contenait que des zéros. On avait deux options pour résoudre ce problème : soit un pouvait implémenter l’instruction nous même dans l’ordinateur, soit on pouvait trouver un autre moyen pour vérifier si l’’accumulateur contenait que des zéros avec les instructions à notre disposition.

Le programme convertit en code machine et inséré dans le fichier .mif :





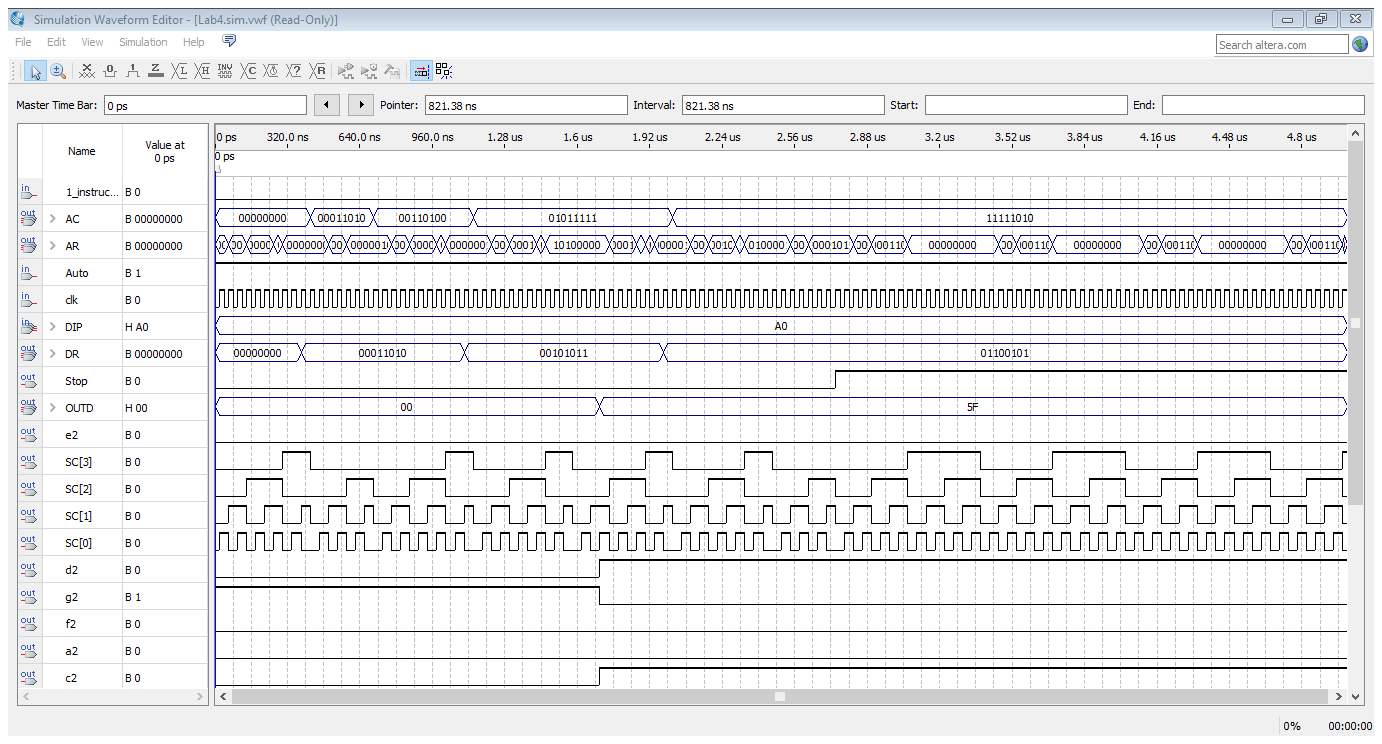
Réalisation réelle

Vérification (simulation réelle & démonstration au professeur)

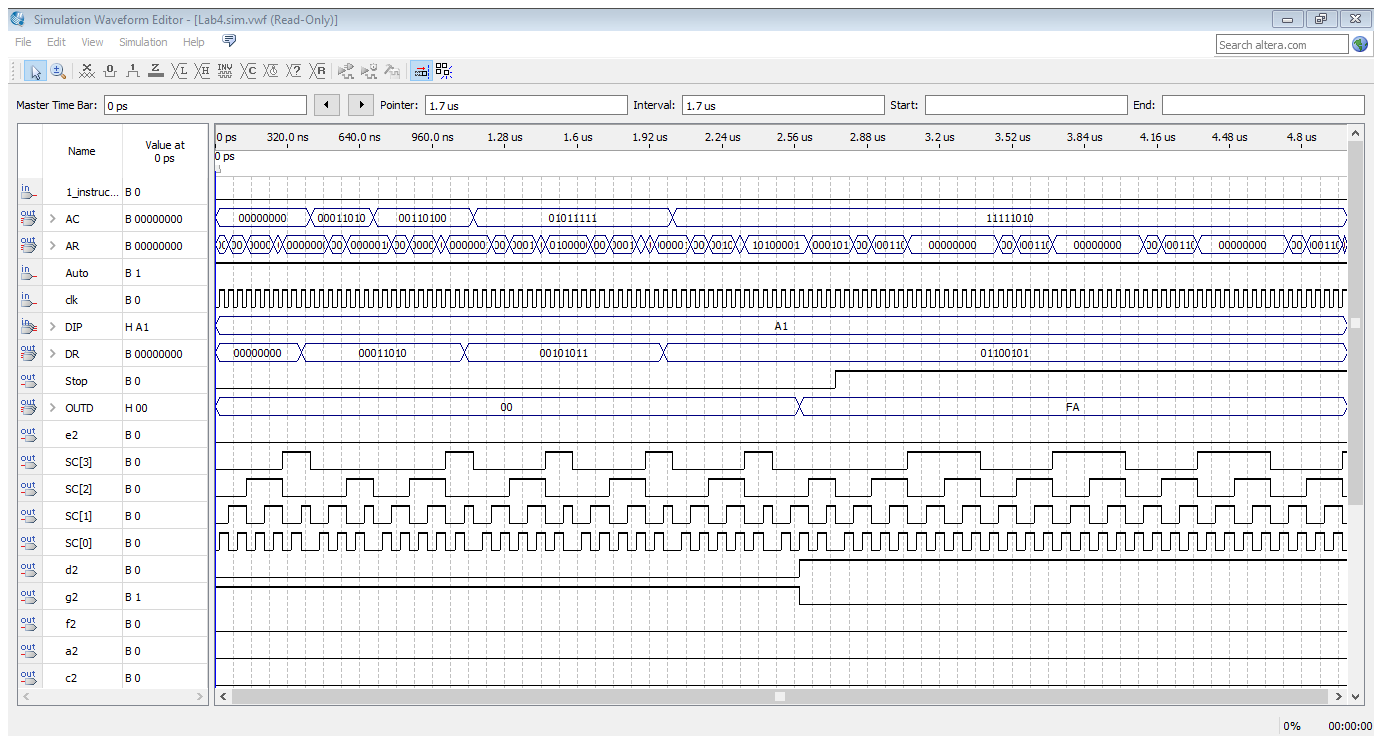
**Partie Conception de l’unité de contrôle**

* Memwrite = T9Y4 + T10Y6
* AR\_LOAD = T0 + T2 + T5 IR6’ + T6IR6’ + T7X2
* PC\_LOAD = T8Y5
* PC\_INC = T2S’ + T5 IR6’S’ + T11Y6 + T12Y6
* DR\_LOAD = T8 (Y0 + Y1 + Y2 + Y3 + Y6)
* DR\_INC = T9Y6
* IR\_LOAD = T3
* AC\_CLEAR = T5 X1 IR0
* AC\_LOAD = T9 (Y0 + Y1 + Y2 + Y3) + T5X1 (IR1 + IR2 + IR3)
* AC\_INC = T5 X1 IR4
* OUTD\_LOAD = T1
* ALU\_SEL2 = T9Y0  + T9Y3 + T5 X1 IR1
* ALU\_SEL1 = T5 X1 (IR1 + IR2 + IR3) + T9Y3
* ALU\_SEL0 = T9Y2 + T5 X1 (IR1 + IR3)
* BUS\_SEL2 = T0 + T9Y4
* BUS\_SEL1 = T2 + T5 + T10Y6 + T0
* BUS\_SEL0 = T8Y5 + T10Y6 + T9Y4
* SC\_CLEAR = T5X1 + T9(Y0 + Y1 + Y2 + Y3 + Y4) + T8Y5 + T12Y6
* HALT = T5 X1 IR5

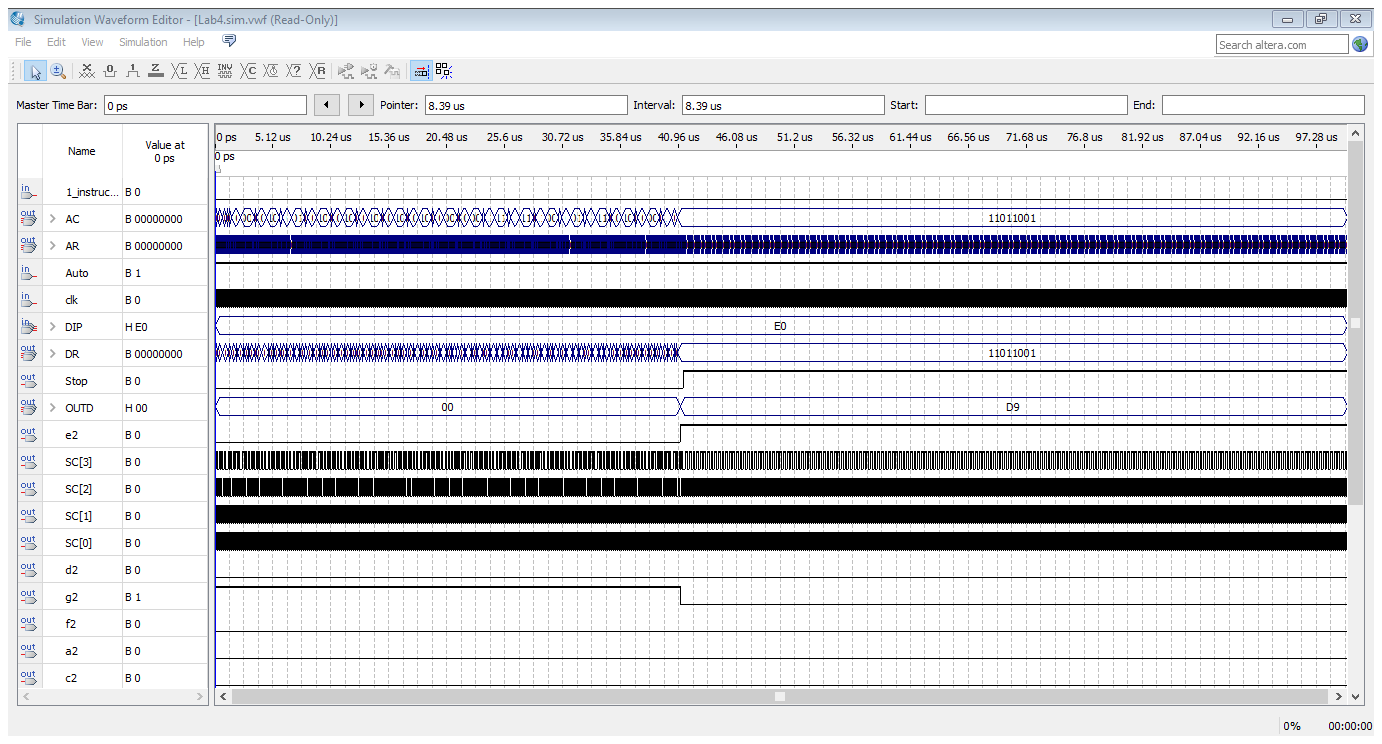
Simulation de la partie conception de l’unité centrale pour DIP A0 :



Simulation de la partie conception de l’unité centrale pour DIP A1 :



**Partie Conception du logiciel**



Discussion

Cette expérience de laboratoire était axée sur la conception de l’ordinateur de base. Il y avait 2 étapes essentiels, la conception du l’unité de contrôle et sont teste puis la conception d’un programme et vérification de ce programme.

Pour concevoir notre unité de contrôle, nous avons extrait les équations des signaux de   
commande. Nous les avons ensuite assemblés dans le diagramme pour pouvoir créer l’unité de   
contrôle. Ensuite, nous devions tester cette partie que nous avons créée, on a d’abord rajouté   
une sortie OUTD dans le fichier global lap3top pour pourvoir examiner les signaux qui passent   
par ce bus. Après cela on a stimulé et programmé sur la carte Altera. Nous avons mis les DIP qui montrent l’adresse à laquelle on voulait accéder et les LEDs à 7 segments nous montraient le   
contenu de la mémoire à l’adresse spécifiée par le DIP.

# Conclusion

Apres cette expérience, nous pouvons conclure que la conception et l’implémentation d’un programme se fait grâce à un ensemble d’instructions et de processus se déroulant avec les registres de base d’un ordinateur, connectés à un système de bus et de la mise au point d’un contrôleur. En effet, ce laboratoire nous a permis de concevoir et de comprendre les différentes étapes d’un programme et pour finir de vérifier les résultats grâce à l’afficheur 7 segments de la carte Altera. Les différentes manipulations que nous avons effectuées nous ont permis d’atteindre cet objectif.

Enfin, les résultats expérimentaux étaient identiques aux résultats théoriques ce qui nous a permis de mettre en évidence leur véracité.