**Lab 4 – BASIC COMPUTER ORGANIZATION**

**CEG2536 A – Architecture des Ordinateurs I**

**Automne 2017**

**École de Science Informatique et de Génie Électrique**

**Assistant de Professeur: Iymen Abdella**

Session de Lab 1

**Charbel Azzi**, 7663780

**Georges Ghossein,** 7653513

Date d’Expérience: 01/12/17

Date de Soumission: 8/12/17

Partie théorique

Introduction du problème ou au laboratoire

**Hardware**

L’unité de contrôle est une partie très importante qui se retrouve dans les ordinateurs de base. Elle est partie du corps de l’ordinateur qui procès l’information et elle contrôle l’ALU, la mémoire et les autres parties des ordinateurs pour répondre à une instruction. Ce laboratoire consiste de crée cette unité de contrôle et assure qu’elle fonctionne avec les autres parties de l’ordinateur telle que la mémoire et l’ALU.

**Software**

De nos jours, Il existe plusieurs types de mémoire telle que les Read Only Memory (ROM), Randomly Accessible Memory (RAM) et plusieurs d’autre. Les mémoires sont une partie fondamentale d’un ordinateur de base. La partie Software du lab consiste à faire la conception d’un type de programme qui va être stocker dans notre mémoire et va diriger l’unité de contrôle pour exécuter le programme.

Discussion du problème

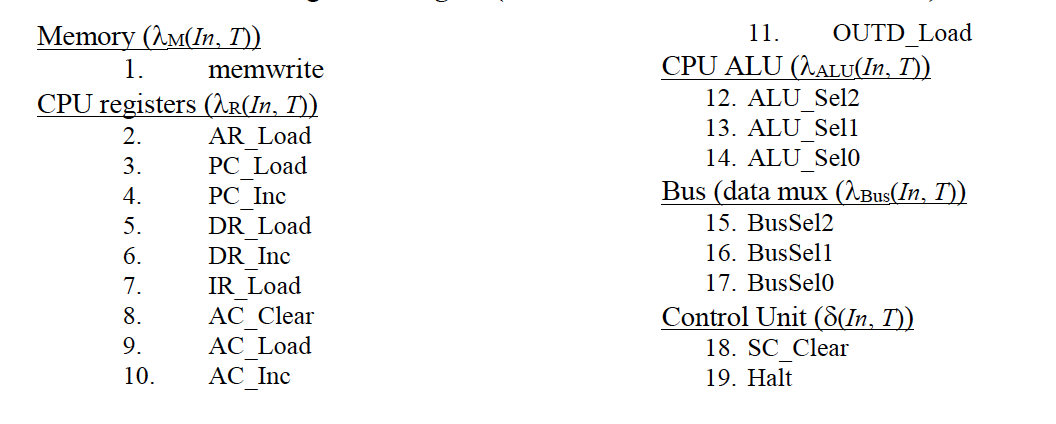
**Hardware**

Le problème, comme introduit dans la partie précédant est de construire une unité de contrôle. Pour savoir comment construire cette unité de contrôle, il est important de savoir qu’elle instruction l’ordinateur pourrait recevoir et comment un ordinateur de base fonctionne.

La majorité partie de l’ordinateur de base a été fournie, donc ce lab va concentrer plutôt sur l’implémentation de l’unité de contrôle qui est notre objectif et non sur l’implémentation global d’un ordinateur de base.

Pour trouver tous les instructions qu’on doit implémenter on doit référencer les tableaux 2, 3, 4 et 5 du pré-lab.

Notre unité de contrôle doit contrôler 19 signaux de contrôle, ces signaux de contrôle peuvent être retrouver ci-dessous :



**Software**

Cette partie comporte d’écrire un programme dans l’adresse mémoire. Du pré-lab, on peut extraire que le programme a pour but d’additionner des nombres déjà stocker dans le mémoire jusqu’à temps que la somme devient 0 et ensuite stocker le dernier nombre utiliser dans la somme.

Les nombre à additionner en ordres sont les suivants : 21, B5, 37, 08, 5C, 84, A1, 1D, 72, FF, F6, 43, 03, A9, D4, 19, 31, D9, 47, 82, 14, 52, 07, CA, 04.

Discussion de la solution algorithmique

**Hardware**

Puisque l’unité de contrôle communique avec plusieurs éléments dans l’ordinateur de base il est très important à trouver chaque instruction qui doit être accomplis. Cette partie, comme mentionner dans la partie ‘Discussion du problème’ peuvent être retrouver dans les tableaux du pré-lab. Cependant, les tableaux ne sont pas assez, il est important de décoder les tableaux. En effet, pour implémenter l’unité de contrôle il faut faire la conception d’un circuit qui sait quand il faut stimuler un des pins des registres pour faire une instruction. Par exemple, disons qu’une instruction indique qu’a T5 on doit faire un clear au registre AC. Notre unité de contrôle doit envoyer un signal au clear pin du AC pour faire l’operation lorsque on est à T5.

On peut décoder les tableaux de la manière suivante : A chaque instruction, donc à chaque ligne, on indique l’état des signaux qui doivent être high (ON) des registres pour accomplir l’opération.

Prenons l’exemple de la première ranger dans le tableau 3 du pré-lab qui indique le suivant :

../../../Screen%20Shot%202017-12-04%20at%209.21.24%20PM.png

Qu’est-ce qu’on peut déduire de cette instruction est que lorsque on est dans l’était T5I on doit stimuler le pin AC\_Clear (qui est partie des signaux de contrôle).

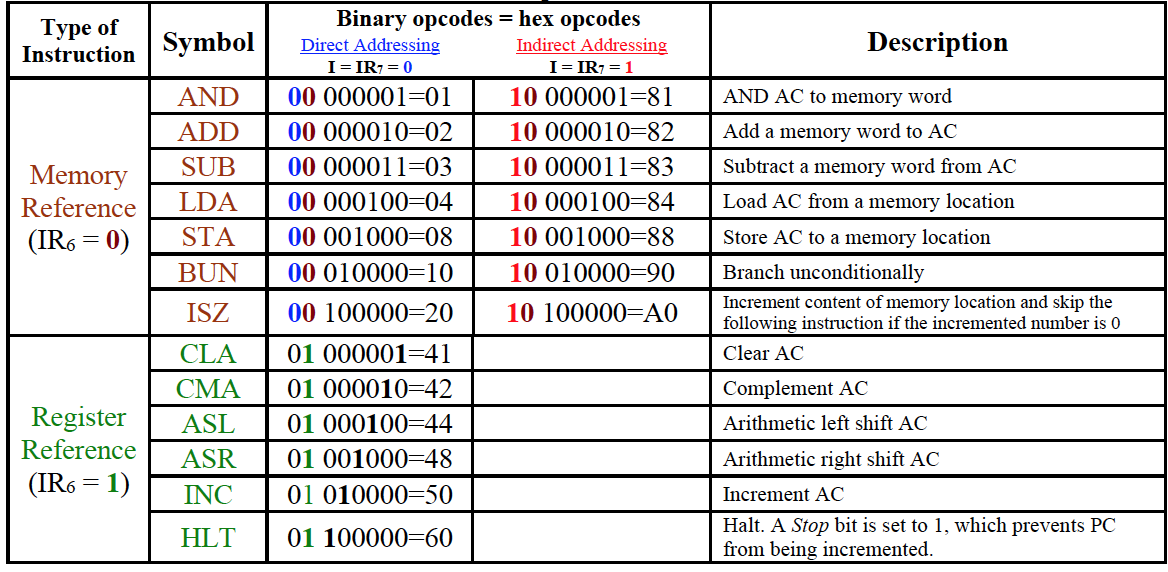
Ensuite, on répète ceci pour chaque opération et on combine tous les possibilités ensemble.

**Software**

Chaque ordinateur peut avoir plusieurs diffèrent type de instructions qu’on peut programmer dans notre mémoire pour faire un programme.

Pour commencer à faire la conception du programme il est important de savoir qu’elle type d’instruction on peut utiliser dans notre mémoire. L’ordinateur de base a plusieurs types d’instruction qui peuvent être démontré dans le tableau suivant :

Tableau 1 : Liste d’Instruction



Une composante importante pour faire la conception de notre programme est de savoir comment implémenter des loops avec ces types d’instruction.

Il est possible d’implémenter une loop en utilisant deux instructions BUN et ISZ. En principe, BUN nous permet de revenir au début de notre sous program qui fait notre addition et le ISZ nous permet de sauter cette opération de Branch lorsqu’on est satisfait ou dans notre cas, lorsque AC est 0.

Partie de conception

Présentation de la méthodologie de conception appliquée à la résolution des problèmes de laboratoire

**Hardware**

Comme démontrer dans la section précédant, on pouvait trouver le circuit combinatoire pour chaque élément en observant les tables et observé chaque instruction. Dans la partie Hardware, il a était aussi important d’implenter une partie dans la memoire qui va en effet envoyer les instructions vers les autres composantes.

Le fichier memoire qui a était utiliser pour implementer la partie hardware est la suivante :

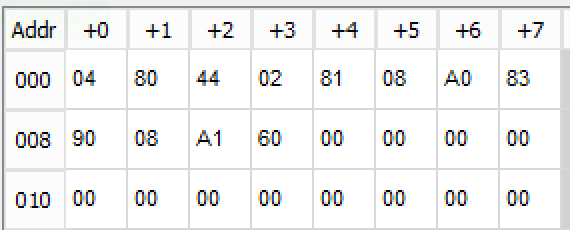




Figure 1: Programme pour la partie Hardware (6.1.3)

***Software***

Puisque le programme nous demande d’additionner plusieurs nombres ensemble, on a pensé d’utiliser les Loop pour concevoir notre programme. Le design a été simple, on entre la loupe, on additionne le nombre prochain dans la liste et on vérifie si AC est 0. Si AC est 0 on skip la Loop et si AC n’est pas 0 on retourne dans la loupe pour additionner le prochain nombre.

Voici le pseudo code pour notre programme :

* Initialiser Pointer (Load et Store)
  + Ce pointer va etre utiliser pour pointer au adresse qui contient les nombres qu’on veut additionner
* Clear AC
* \*LOOP Commence\* Additionner le nombre qui se retrouve dans le pointer
  + Incrémentez pointer
  + Check si AC est 0, si oui skip, sinon retourne a la loupe
* HALT

Discussion sur les composantes utilisée

**Hardware**

Dans la partie hardware les composantes qu’on a utilisées sont les tableaux du pre-lab. Ces tableaux on était les plus importantes puisqu’on a dérivé notre circuit combinatoire à leurs aides. De plus, d’autres composantes utilisées pour nos circuits combinatoires sont les portes logiques utilisées pour implémenter le circuit combinatoire dont les Portes : OR, AND, NOT. On a aussi servi du GND et VCC.

**Software**

Les composantes utilisées pour le développement du software étaient les instructions que l’ordinateur de base peut accepter. Ces instructions peuvent être retrouvé dans le tableau 1.

Discussion de l’outil

Les outils utiliser dans ce laboratoire était :

* Quartus 2
* Board Altera

Ceci est les seuls outils qu’on a utilisés.

Discussion de la solution réelle

**Hardware**

La solution réelle pour la partie hardware est composée de plusieurs sous blocks qui fait parties d’un grand block. En réalité, si ceci avait été implémenter avec des vraies composantes on aurait beaucoup plus de fils (wire) cependant, on a utilisé le pouvoir du compilateur quartus qui lui-même sait lié des fils avec les mêmes noms.

Aussi, qu’est-ce qu’on a fait pour diminuer le Delay le plus qu’on peut est qu’on a utilisé seulement une porte NOT pour un entrée qui a était utilisé plusieurs fois pour diminué le Delay le plus que possible. En autre mot disons que T5’ a été utilisé à plusieurs entrées dans les sous blocks, on a implémenté cette fonction seulement une fois et on a référencer le fils avec le nom du fils approprié.

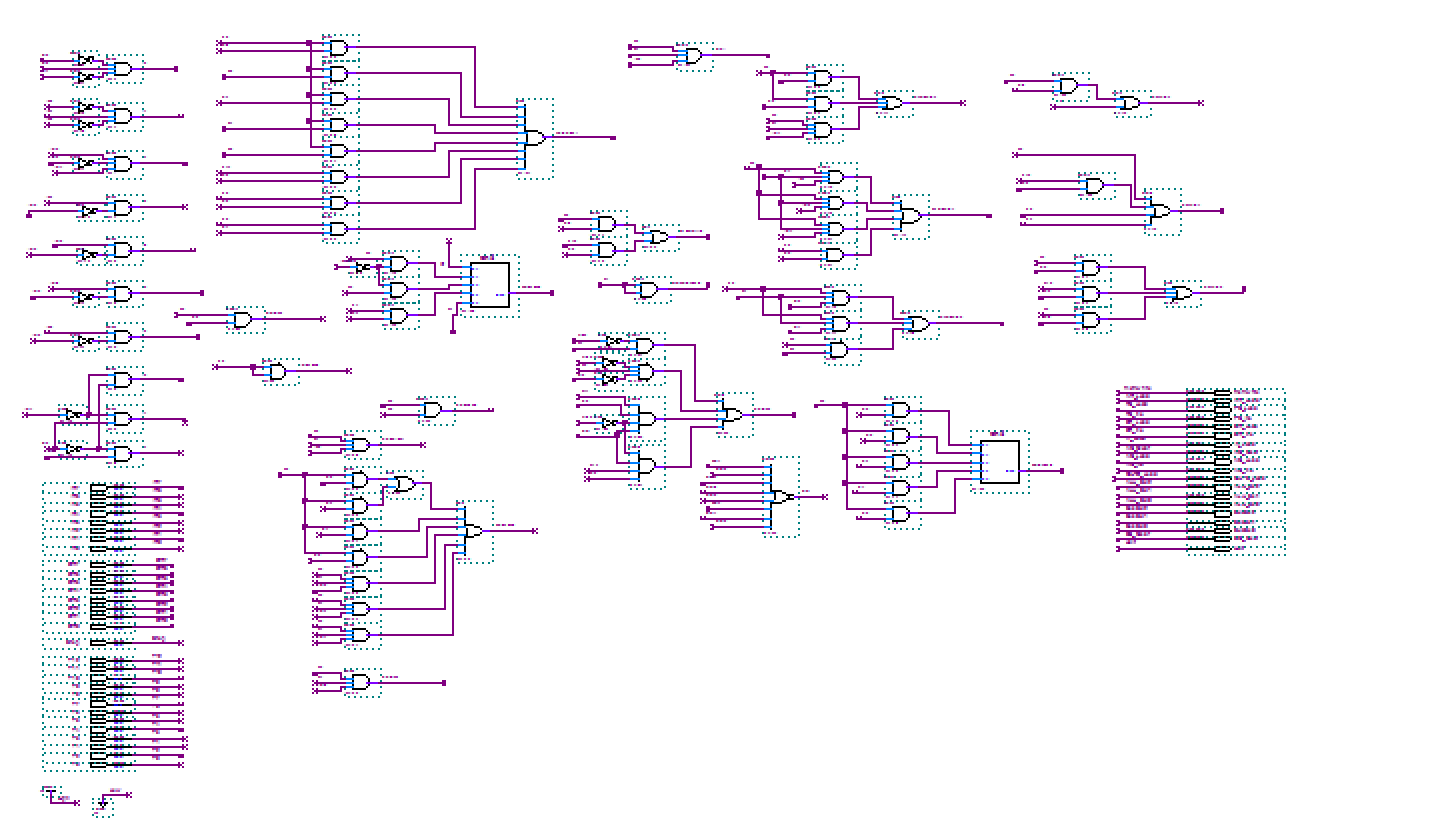
****

Figure 2 : Schéma du fichier Lab 3 controller (6.1.2)

**Software**

La solution réelle du software à été seulement d’implémenter notre pseudo code dans notre mémoire.

Voici le fichier mémoire qui implémente notre programme :

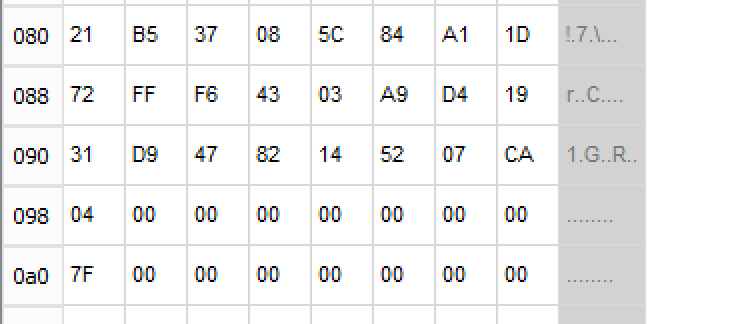
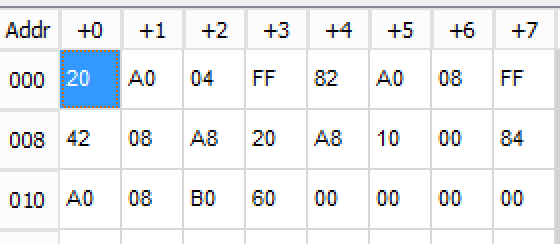


Figure 5 : Programme pour la partie software (7.2.1)

Par la suite, nous avons effectué une simulation afin d’observer le fonctionnement du programme et assurer que la schématisation de l’unité de contrôle s’est bien dérouler. Nous avons effectué deux simulations où nous avons comparé le résultat suite au changement de l’entrée DIP soit de A0 à A1.

PARTIE SIULATION

Pour la partie simulation, le TA à noter que les 4 points sont suffisant :

* le prélab (pour les deux parties: section 5.1, et section 7.1)
* un capture d'écran pour la conception de ton circuit (lab3controller)
* deux captures d'écran pour le test de ton circuit (que moi ou Steve ont vérifié, section 6.1.1 étape 7). Un quand DIP = A0, et l'autre quand DIP=A1.
* capture d'écran pour le test de la programme que vous avez conçu pour la section 8.6.2

Les prélab s retrouve dans la section ANNEXE.

Pour les capture ecran pour prouver que la solution est fonctionelle,ils se retrouve dans les figure suivante :

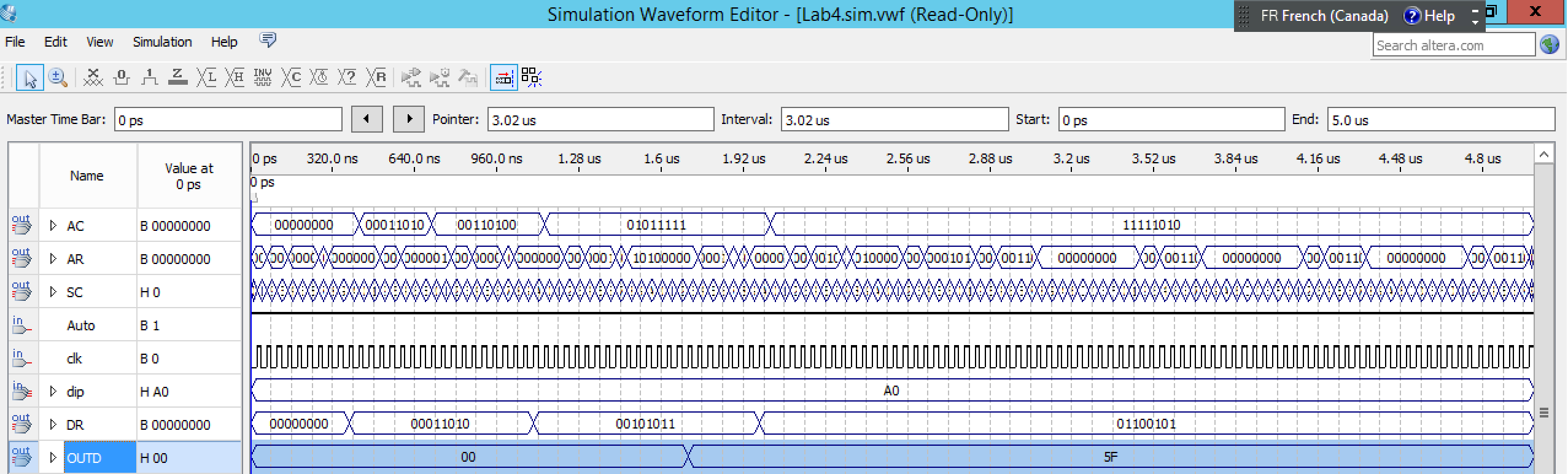


Figure 3 : Résultat obtenu suite à la simulation où DIP est à A0 (6.1.7)

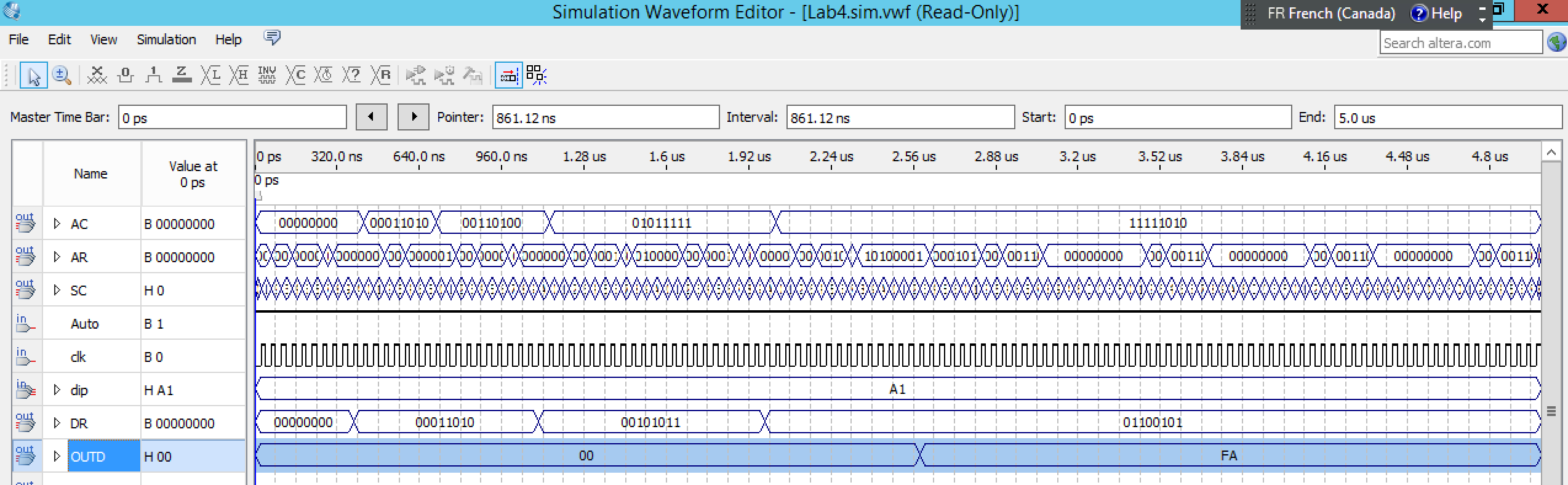


Figure 4: Résultat obtenu suite à la simulation où DIP est à A1 (6.1.7)

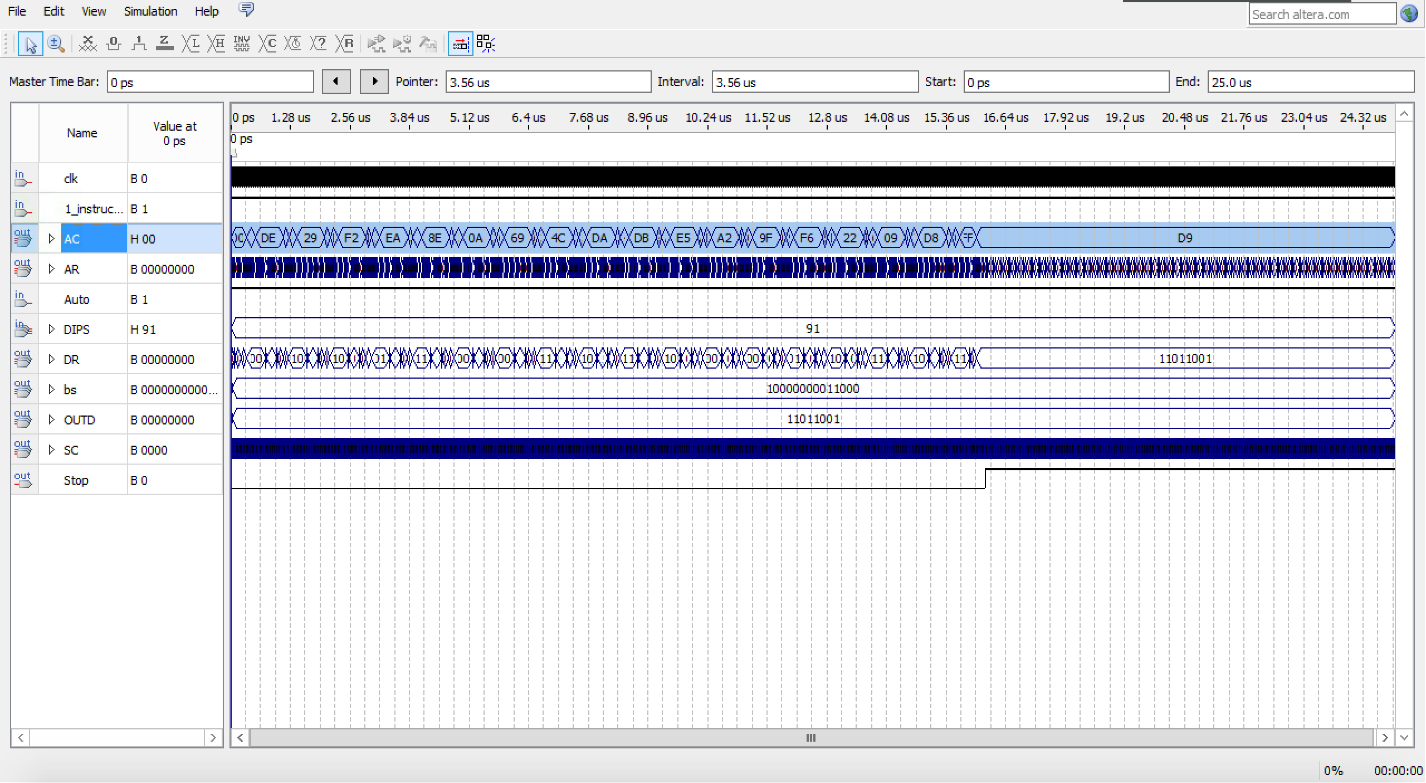
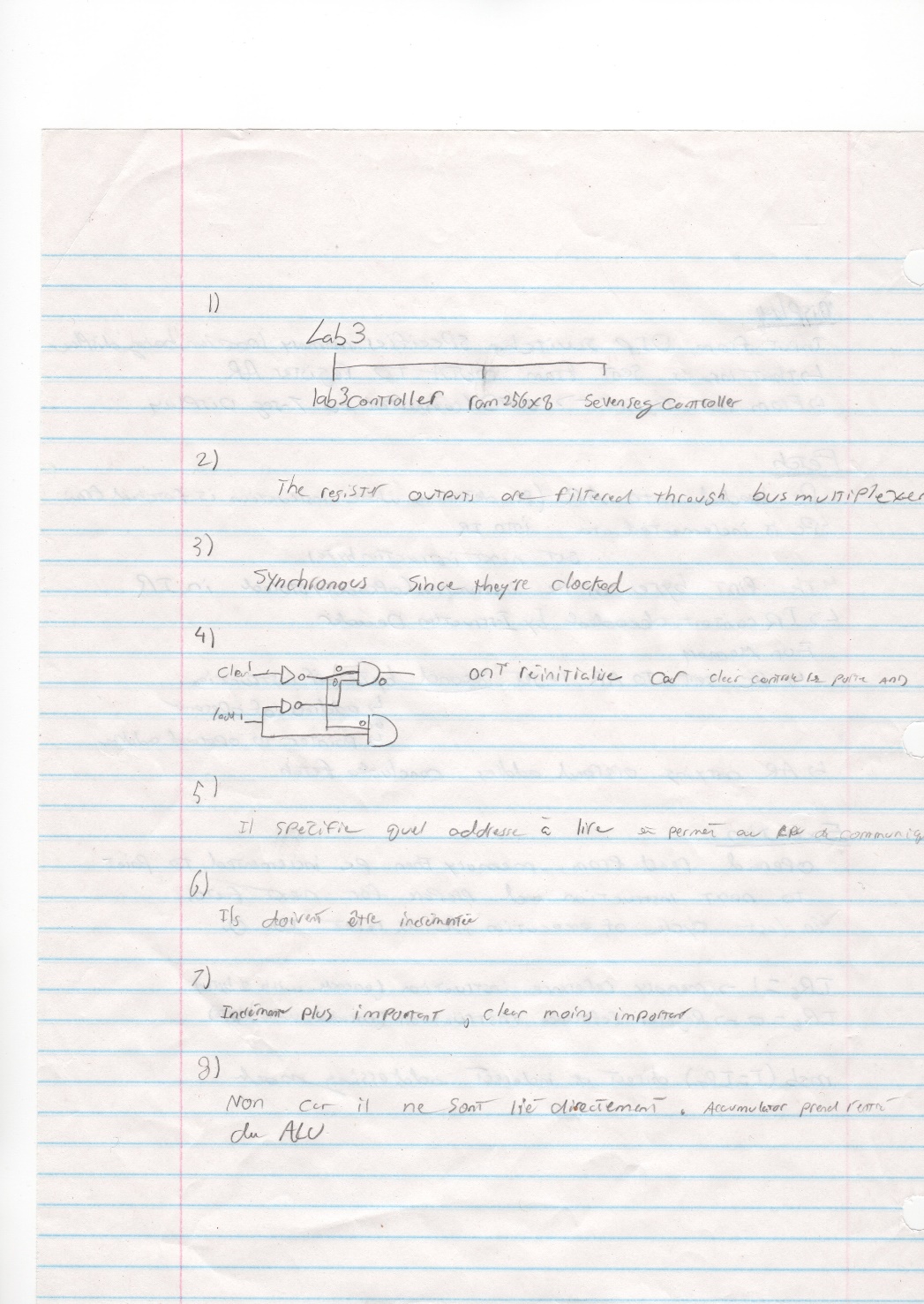


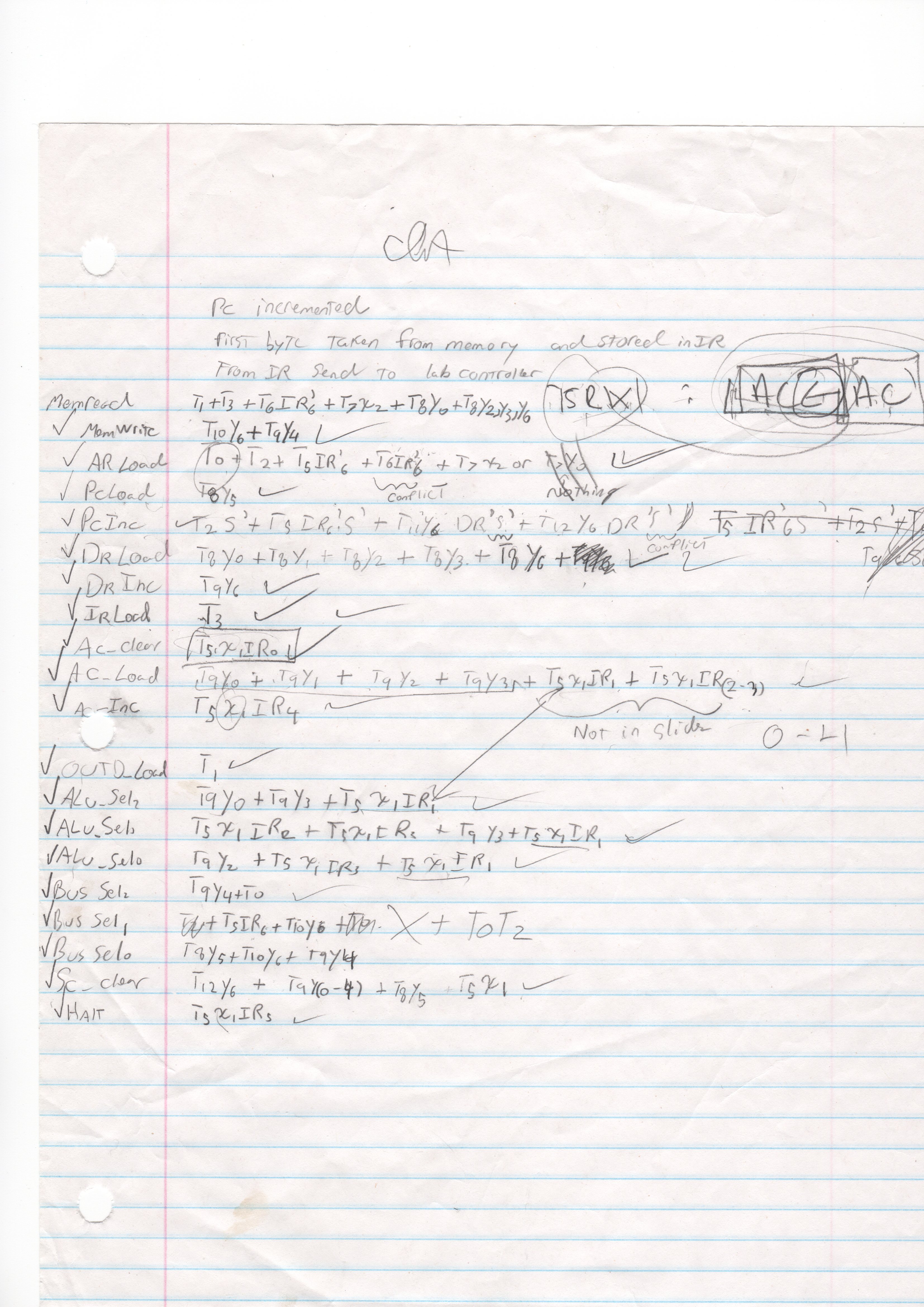
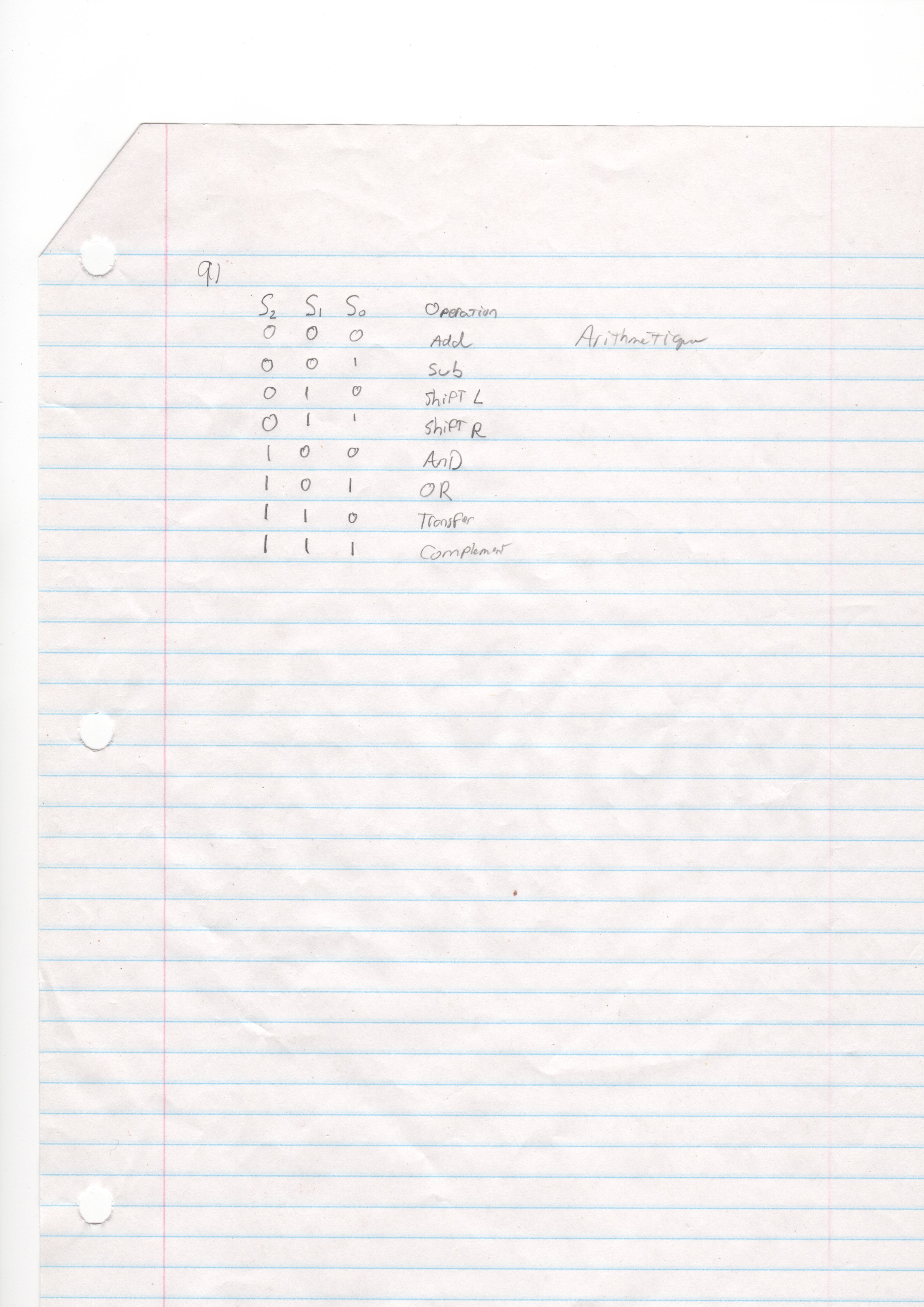
Figure 5 : Résultat obtenu suite à la simulation où DIP est à A1 (6.1.7)

A partir de ces images de solutions, on est confident que notre solution est fonctionelle.

Conclusion et Discussion sur les points difficiles

Pour conclure, lors de cette expérience nous avons schématisé l’unité de contrôle retrouvé dans un ordinateur de base pour la première partie du lab. Nous avons aussi effectué des simulations ainsi que la programmation sur la carte altéra afin d’observer son fonctionnement et d’approfondir nos connaissance sur le sujet. De plus, nous avons appris a programmé en assemblage où la liste d’opération utilisé était fourni. Ayant pour but d’additionner consécutivement des numéros d’une séquence fournie, nous avons implémenté le programme dans notre ordinateur de base et observer le résultat, soit le dernier numéro ajouté lorsque la somme devient égale à 0. Les points difficiles du laboratoire on était durant la construction de la partie hardware du lab. La partie software n’était pas trop difficiles puisqu’on avait seulement à construire un programme dans la composantes mémoires. La partie difficile qu’on a trouvée a été de trouver le circuit combinatoire qui représente chaque broche (pin). Puisqu’il y avait plusieurs tableaux ont considéré et plusieurs instructions qui se répétaient il a été très difficile à savoir dans note pré-lab si on a bien conçu chaque sous-block de notre Controller ou non.

ANNEXE A : Prelab Hardware



ANNEXE B : Prelab Software

