Exercice 1 :

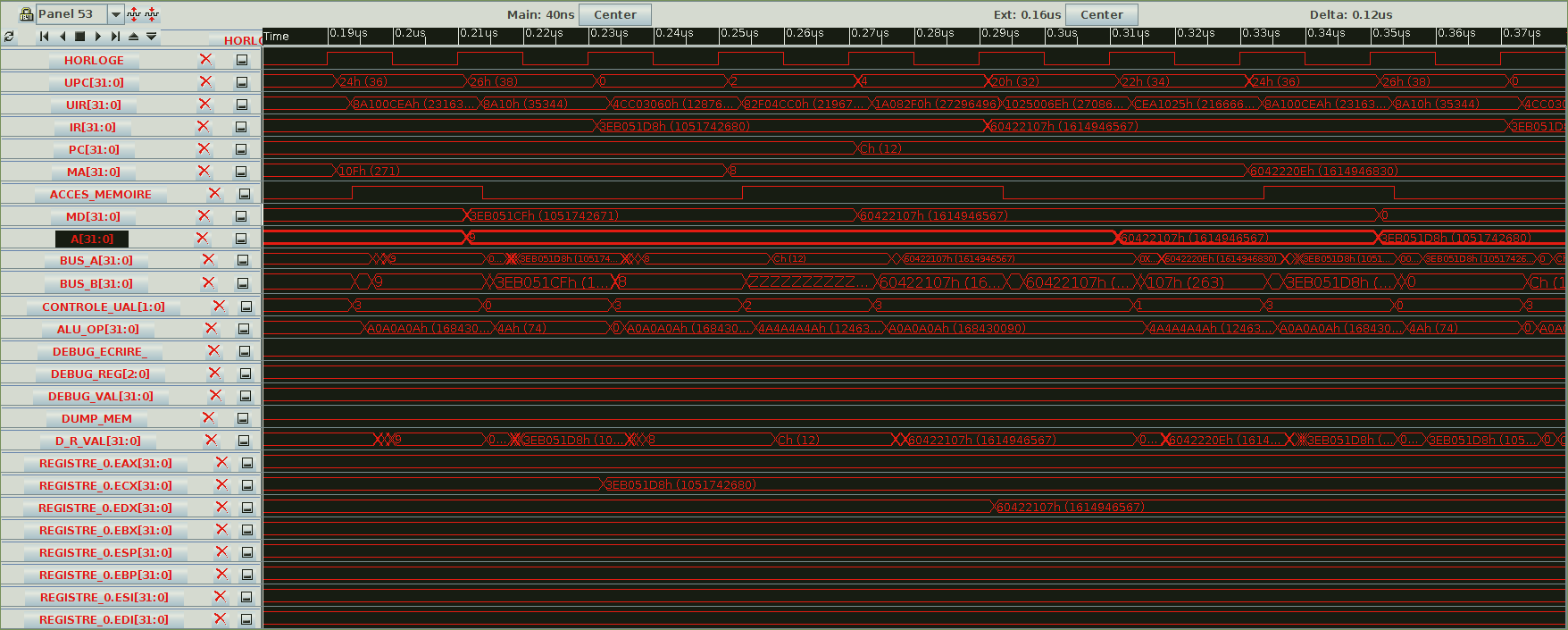
1-

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RTN concret | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | hexa |
| MA<-PC; | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0x3060 |
| MD<-M[MA] :  PC<-PC+4; | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0x4CC0 |
| IR<-MD; | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0x82F0 |

2-Execution d’une instruction générique :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RTN concret | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | hexa |
| A<-R[IR<16..12>]; | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0x006E |
| MA<-A+IR<11..0> | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0x 1025 |
| MD<-M[MA] :  A<-R[IR<21..17>]  (a prend la valeur du registre rb)  Qui est edx | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0x0CEA |
| R[IR<26..22>]<-MD oper A  Ra (ecx) | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0x 8A10 |

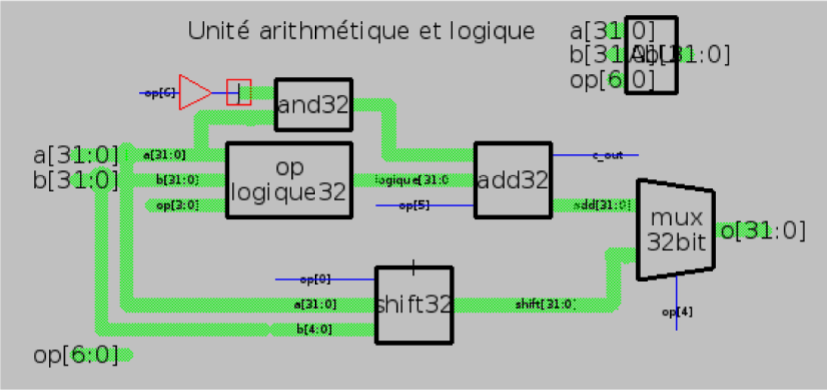
3-Simulation

Le résultat obtenu est 1051742680 .Il s’agit de l’addition entre le contenu de la mémoire de donnée et le contenu du bus A.

A la troixieme instructiion (PC vaut 8),il est demande a UAL d’effectuer une operation d’addition(add) entre le contenu de la mémoire (MD qui vaut «’ 105174271’) de donnée et le contenu du bus A (qui vaut ‘9’).Le resultat est coherent avec les données et l’instruction recu.

Ce résultat est placé dans le registre ecx (qui symbolise R[IR<26..22>]).

4-Operation nand



Pour identifier l’octet qui sera place dans le fichier ‘opalu’

Nous avons déterminé les sept bits que comportent le vecteur ‘op’

Le bloc op logique 32 permet une opération logique sur a et b.En vue d’effectuer l’opération a nand b, les 4 premiers bits du vecteur op ont été déterminé en passant par la table de vérité NAND

|  |  |  |
| --- | --- | --- |
| a | b | Op[0 :3] |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Le bit op[0] reste a 1 car il a été déterminer par la table de verité.

Le bit op[6] =0 afin d’obtenir une sortie nulle sur 32 bits .

Le bit op[5] =0 afin que le module add 32 effectue une opération qui conservera la valeur logique[31 :0](l’addition se fera entre une opérande significative et deux opérandes nulles).

Le bit op[4]=0 afin que le multiplexeur mx32 aiguille sur la sortie finale de ALU la sortie du bloc add32 (qui est la valeur contenu dans le vecteur logique[31 :0])

Op[6 :0]=0001110

L’octet placé dans le fichier ‘opalu’est 07h(0000 1110 en binaire)par l’ajout d’un bit 0 a op[6 :0] en vue d’avoir un bit sur 8 octets.



La valeur obtenu suite a une opération nand est

E5DFBE77h(385664575 en base décimal).

Trois cycles après l’instruction add ,à la sixième instruction (PC vaut 16),il est demande a UAL d’effectuer une operation logique NAND entre le contenu de la mémoire (MD qui vaut «’ 9B63E3389h’) de donnée et le contenu du bus A (qui vaut ‘3EB051D8h ’).Le resultat est coherent avec les données et l’instruction recu.

Ce résultat est placé dans le registre edx .

5-comprehension

a-

les bits pairs sont seter a 1;on a donc 14,12,10,8,6,4,2,0 qui sont actifs

on ecrit pc ,ma

,md,ecrit dans la memoir,

On fait une addition,on ecrit dans la banque de regisre on selectionne le regidate rc en lecture

On met la constance sur le bus

b-

L’avantage est que l’on a accès aux données ,que l’on peut traiter les données ainsi que sauvegarder le résultat dans la mémoire dans un même cycle .

Nous avons utilisé cet avantage lorsque nous avons utilisé la micro-instruction MA<-A+IR<11..0> :

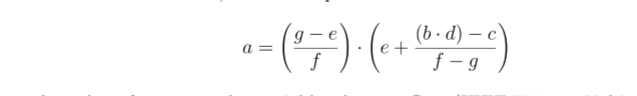
lire sur le bus la constante, effectuer l’addition et sauvegarder le résultat dans la mémoire sur un même cycle.

c-

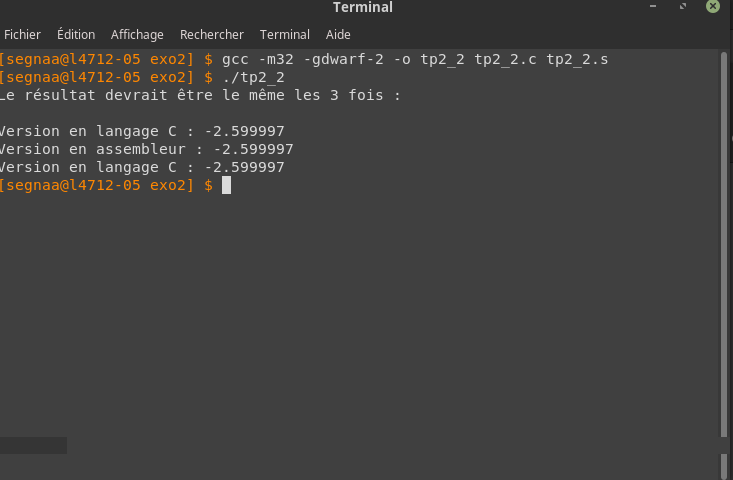
Oui, car on peut conserver une seule fois le résultat de l’UAL pour refaire une opération nécessitant ce résultat.

Exercice 2 :

Pour cet exercice il nous est demandé d’écrire en assembleur l’expression suivante :

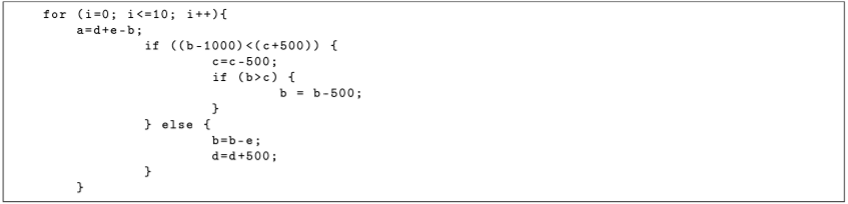


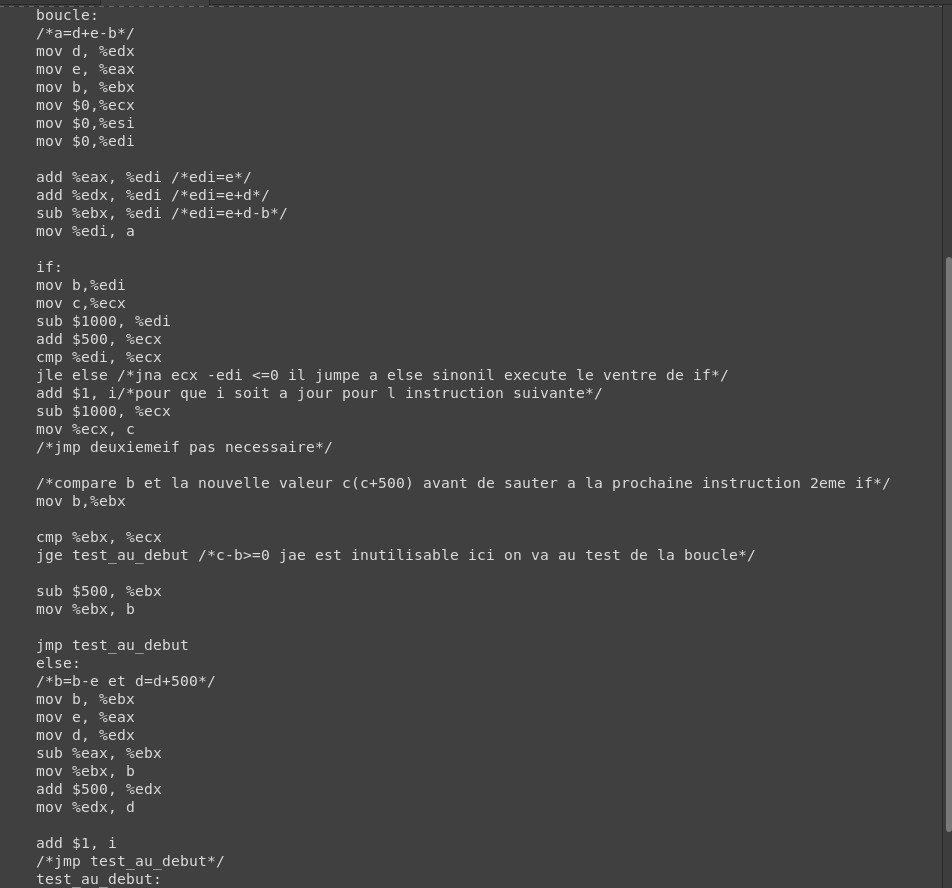
Voici une capture d’écran du code écrit en assembleur : 

Voici une capture ecran de la sortie du terminal :

Exercice 3:

Pour cet exercice il nous est demandé de transcrire en langage assembleur le code suivant :



Voici une cature d’écran d’une partie du code écrit en assembleur :

Voici une capture d’écran de la sortie du terminal :