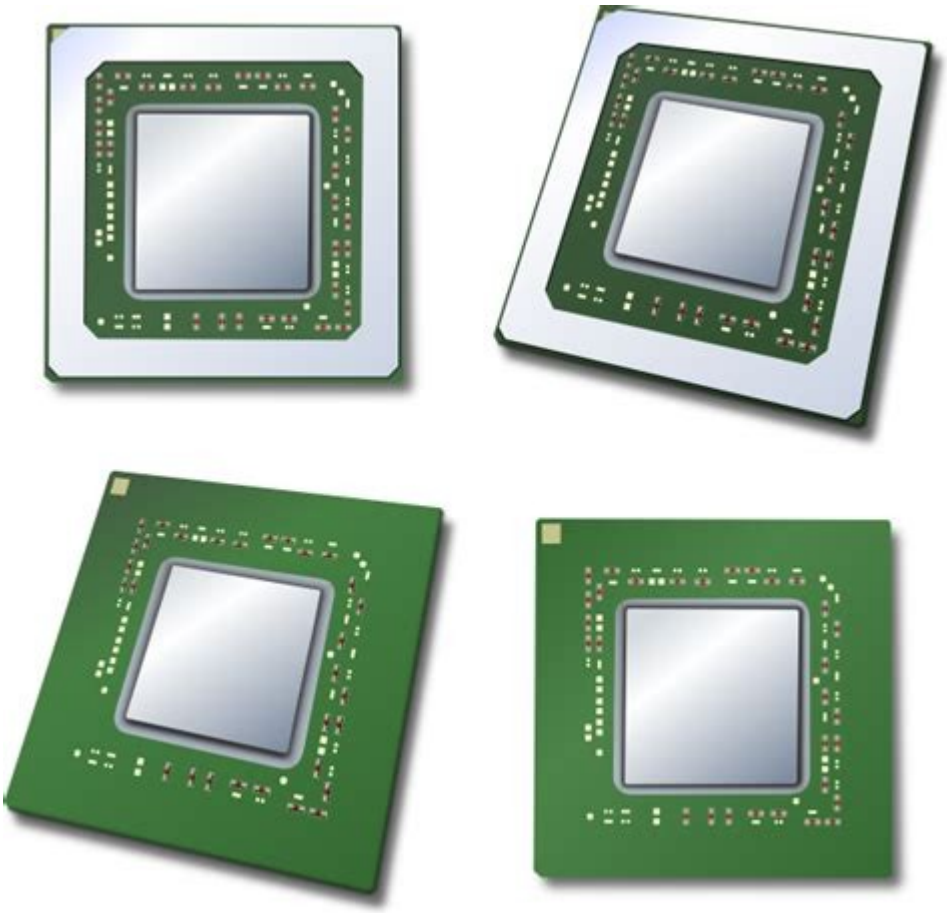
Rendu Microprocesseur



SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS
DEPUIS 1257

POLYTECH
SORBONNE

Table des matières

Introduction

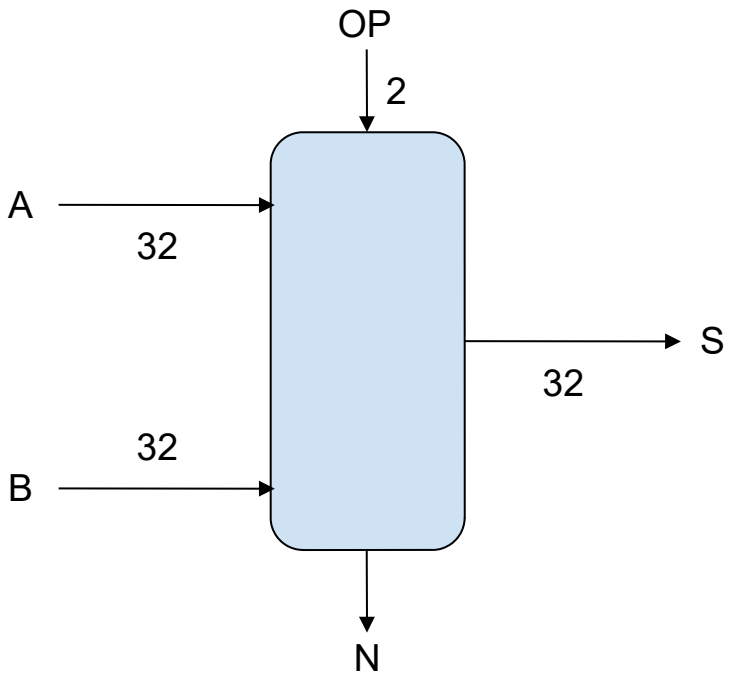
Pour effectuer les opérations arithmétiques et logiques, tout système a besoin d'un processeur. En 1971, Marcian Hoff a inventé le microprocesseur, un microprocesseur désigne un processeur qui possède des composants électroniques suffisamment miniaturisés pour pouvoir tenir dans un seul circuit intégré.

Un microprocesseur est caractérisé par les fonctions suivantes :

- Le jeu d'instructions qu'il est capable d'exécuter, pouvant aller de dizaines à des milliers d'instructions différentes.
- La complexité de son architecture qui se mesure par le nombre de transistors présents : plus ce nombre est élevé, plus la complexité des tâches à traiter peut augmenter.
- La vitesse de son horloge qui dicte le rythme de travail.
- Enfin, le microprocesseur se caractérise par le nombre de bits qu'il peut traiter (4 à ses débuts, 128 en 2011).

Pour le cours d'architecture des processeurs, nous avons travaillé dans la conception d'un microprocesseur monocycle. Dans la suite, nous allons décrire chaque une des entitées créés pour le microprocesseur pour expliquer son fonctionnement.

Unité Arithmétique Logique 32 bits



Entrées :
- **OP** : Il choisit l'opération voulue sur 2 bits
- **A**, **B** : Signaux sur lesquels on applique l'opération sur 32 bits

Sorties :
- **S** : Résultat de l'opération sur 32 bits en sortie
- **N** : Indicateur du signe, 1 si négatif, 0 sinon sur 1 bit

Fonctionnement / Description:
Ce module traite les données de la manière asynchrone et il nous permet de choisir parmis 4 opérations à appliquer aux entrées A et B.

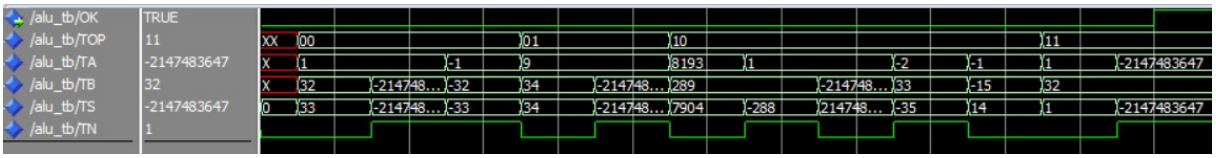
Opérations en fonction de OP :

OP	Type d'opération	Résultat (**S**)
00	**SOMME**	**A + B**
01	**B**	**B**
10	**SOUSTRACTION**	**A - B**
11	**A**	**A**

Banc de test

Pour valider son fonctionnement, nous avons fait une banc de tests faisant les tests suivants :

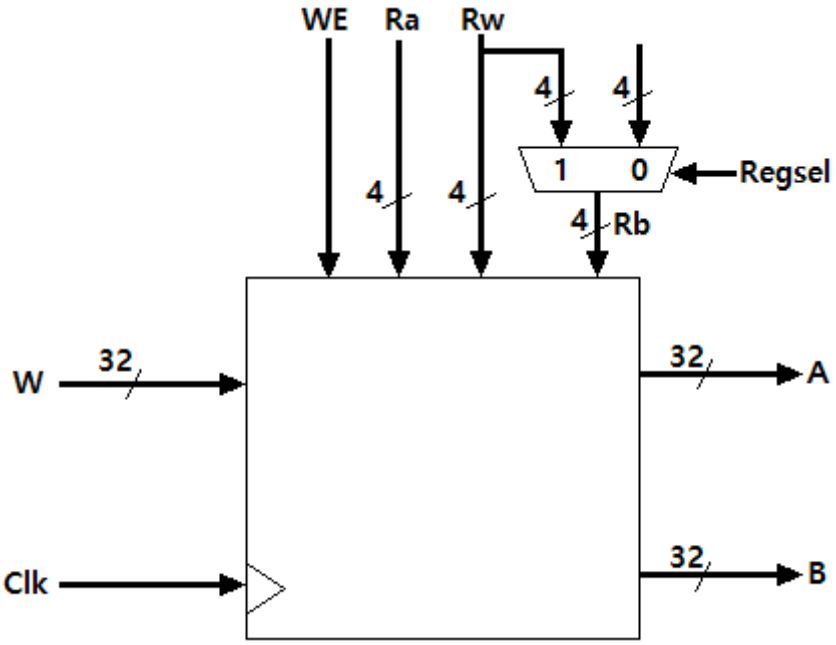
Résultat de la simulation



Pour valider son fonctionnement, nous avons fait un Test Bench faisant les tests suivants :

Opération	A	B	S	N	Vérification
+	1	32	33	0	A positif + B positif
+	1	-2147483616	-2147483615	1	A positif + B négatif
+	-1	-33	-34	1	A négatif + B négatif
B	9	34	34	0	B négatif
B	9	-2147483614	-2147483614	1	B négatif
-	8193	289	7094	0	A positif - B positif = S positif
-	1	289	-288	1	A positif - B positif = S négatif
-	1	-2147483359	2147483360	0	A positif - B négatif = S positif
-	-2	33	-35	1	A négatif - B positif = S négatif
-	-1	-15	14	0	A négatif - B négatif = S positif
A	1	32	1	0	A positif
A	-2147483647	32	-2147483647	1	A positif

Banc de registres



Entrées:

- **Clk** : L'horloge synchronisé au système, sur 1 bit
- **WE** : **W**rite **E**nable. L'écriture est autorisé si le signal est 1, sur 1 bit
- **W** : Les données à écrire dans la banc des registre, sur 32 bit
- **Rw** : L'adresse d'un registre à écrire les données **W**, sur 4 bit
- **Ra** : L'adresse d'un registre à lire, sur 4 bit
- **Rb :** L'adresse d'un registre à lire, sur 4 bit
- **RegSel** : Le signal de commande du multiplexeur pour choisir l'adresse du bus B.

Sorties:

- **A** : Les données lues du registre à l'adresse **Ra**, sur 32 bit
- **B** : Les données lues du registre à l'adresse **Rb**, sur 32 bit

Fonctionnement :

La banc de registres se fait en 16(= 2^4) registres de 32 bits. Donc, la taille disponible est 16 mots. Ces registres sont adressés par les entrées d'adresse (**Rw**, **Ra** et **Rb**). **Rw**, **Ra** et **Rb** sont respectivement associé aux entrée/sorties bus **W**, bus **A** et bus **B**.

La lecture est asynchrone (combinatoire et simultanée). Les sorties **A** et **B** donnent immédiatement les valeurs actuelles de registres à leurs propres adresses(respectivement, **Ra** et **Rb**).

L'écriture est synchrone sur le front montant de l'horloge, **Clk**. Si le signal **WE** est 1, les données passées par l'entrée **W** sont écrites au front montant de l'horloge. Si le signal **WE** est 0, aucune opération d'écriture n'est permise. Donc le signal **WE** est prioritaire.

Description :

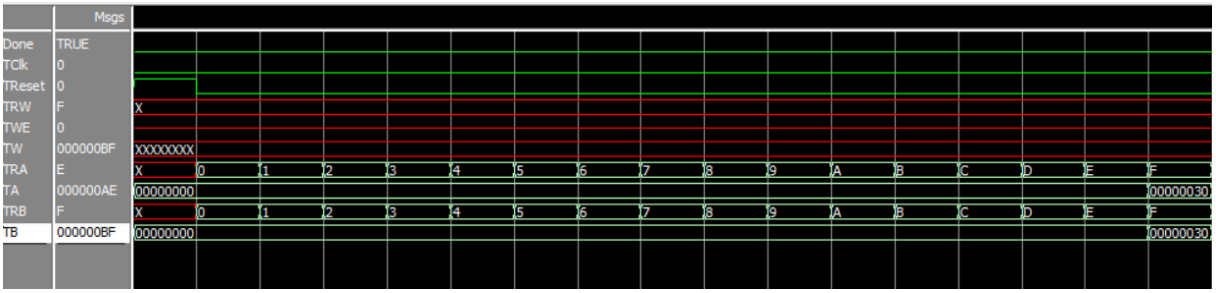
La banc de registres permet un accès rapide à l'unité arithmétique logique. Les opérandes et le résultat de l'UAL sont stockés ici. La taille de la banc de registres est relativement faible, leurs valeurs sont donc facilement écrasées par des nouvelles valeurs.

Banc de test

Pour valider son fonctionnement, nous avons fait une banc de tests faisant les tests suivants :

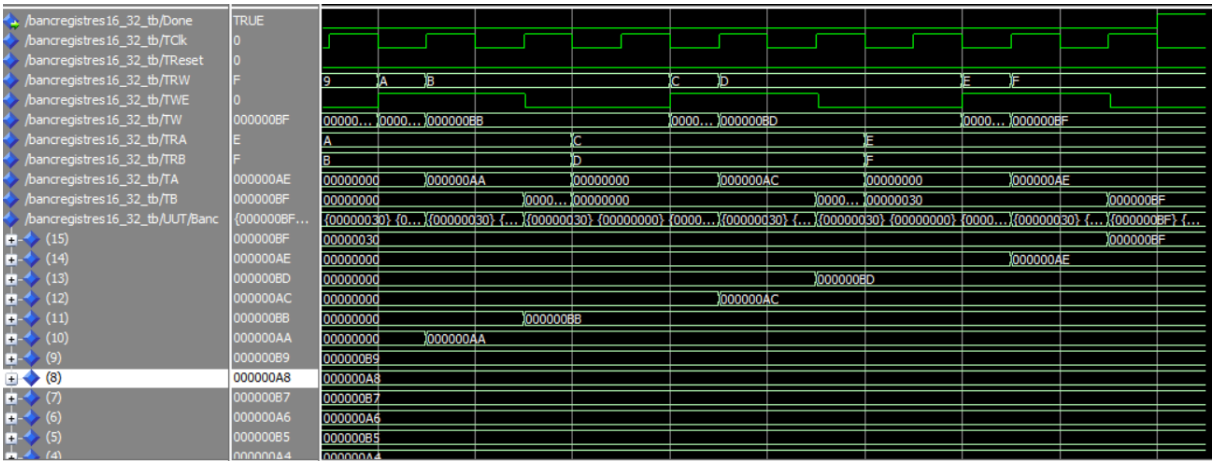
Résultat de la simulation

Des tests de l'initialisation et la lecture:



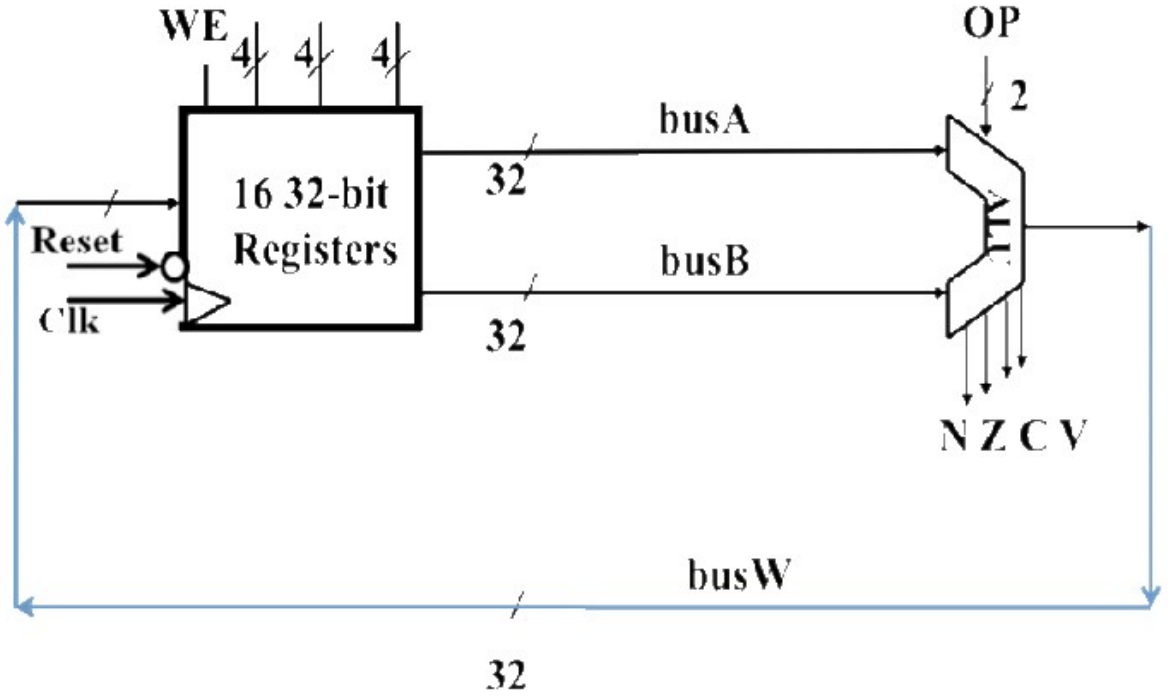
Les valeurs dans tous les registres sont 0 sauf le registre à l'adresse 0xF. La valeur du registre à l'adresse 0xF est 0x30.

Des tests de l'écriture et la lecture:



L'écriture est synchrone. La valeur 0xA +[Rw] est stockée dans les registre de l'adresse paire et la valeur 0xB +[Rw] est stockée dans les registre de l'adresse impaire. L'écriture peut être vérifiée dans le tableau de registres **Banc.**

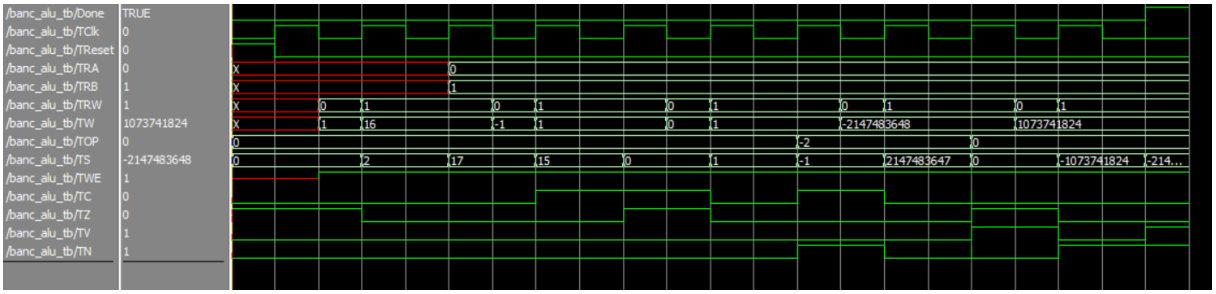
Banc de registre + UAL



Banc de test

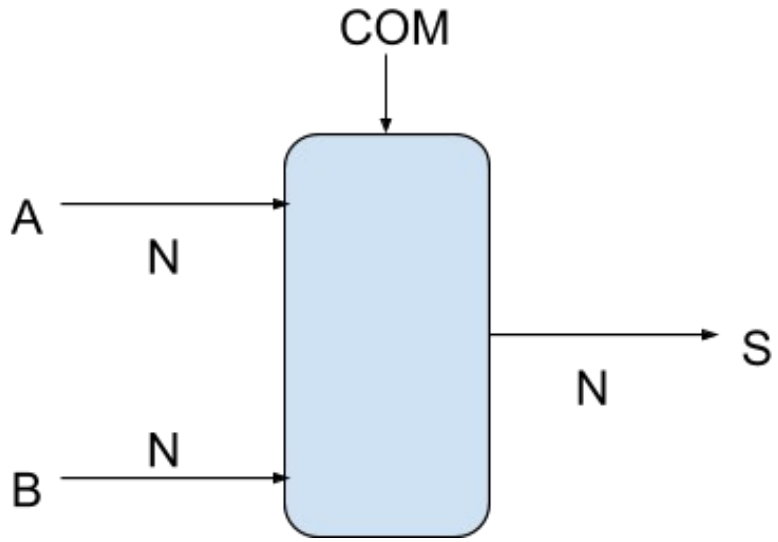
Pour valider son fonctionnement, nous avons fait une banc de tests faisant les tests suivants :

Résultat de la simulation:



Cf. banc_alu_tb.vhd pour les détail du résultat.

Multiplexeur 2



Générique :

- **N** : Fixe la taille des données en entrée et sortie, entier

Entrées :

- **COM** : Il choisit la sortie voulue sur 1 bit
- **A**, **B** : Signaux sur lesquels on travail sur N bits

Sorties :

- **S** : Sur N bits en sortie

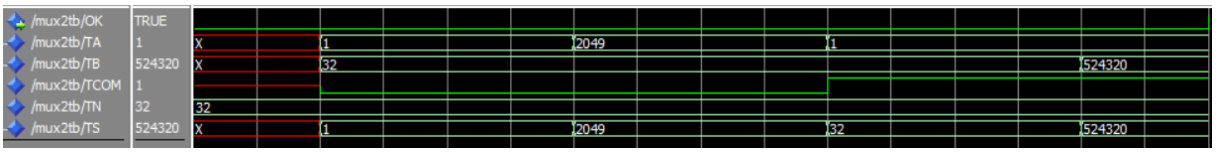
Fonctionnement / Description:

Ce module nous permet de choisir A ou B avec une commande, COM = 0 S <= A et COM = 1, S <= B.

Banc de test

Pour valider son fonctionnement, nous avons fait une banc de tests faisant les tests suivants :

Résultat de la simulation



A	B	COM	S	N
1	32	0	1	32
2049	32	0	2049	32

1	**32**	**1**	**32**	**32**
1	**524320**	**1**	**524320**	

Extension de signe

Générique :

- **N** : Fixe la taille des données en entrée et sortie, entier

Entrées :

- **A** : Signal sur lequel on travail, sur N bits

Sorties :

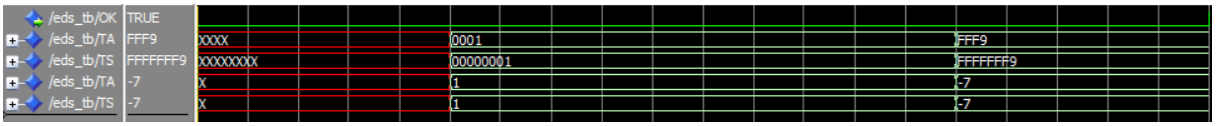
- **S** : Sur 31 bits en sortie

Ce module nous permet mettre A sur 32 bits en tenant compte du signe de A.

Banc de test

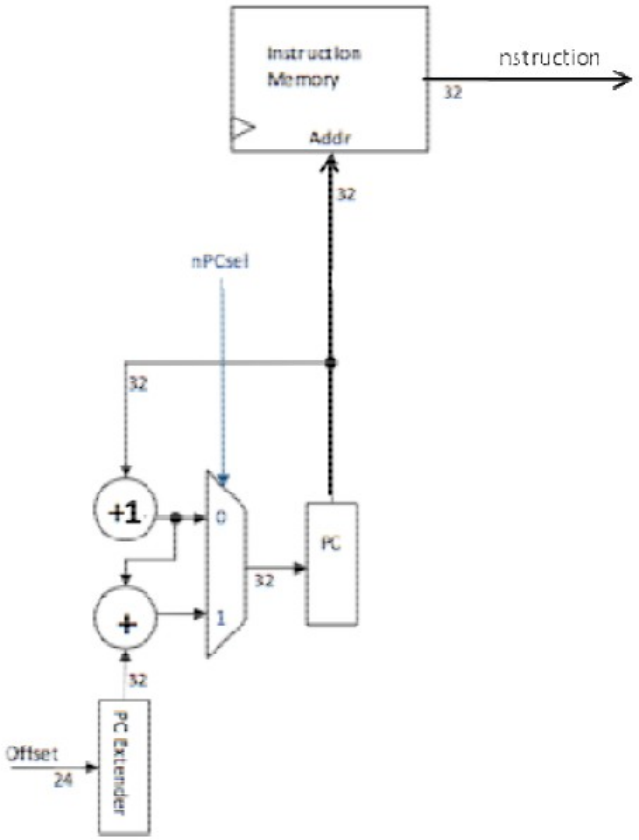
Pour valider son fonctionnement, nous avons fait une banc de tests faisant les tests suivants :

Résultat de la simulation



A	N_A	S	N_S
1	**16**	**1**	**32**
-7	**16**	**-7**	**32**

Unité de Gestion des Instructions



Entrées :

- **Clk** : Signal d'horloge, sur 1 bit
- **Reset** : Signal de Réinitialisation, sur 1 bit
- **Offset** : Offset de l'élément qu'on veut obtenir dans la mémoire, sur 24 bit
- **nPCsel** : Choix de Offset ou PC + 1 comme indice qu'on veut obtenir dans la mémoire, sur 1 bit

Sorties :

- **Instruction** : Sur 31 bits

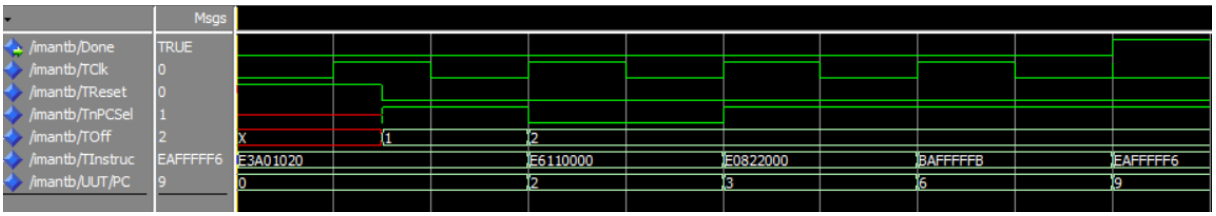
Fonctionnement / Description:

Ce module cherche une instruction dans la mémoire d'instructions en fonction d'un indice. L'indice est PC. PC s'incrémente de 1 à chaque coup d'horloge si nPCSel = 0, si nPCSel = 1 alors il s'incremente de offset plus PC + 1.

Banc de test

Pour valider son fonctionnement, nous avons fait une banc de tests faisant les tests suivants :

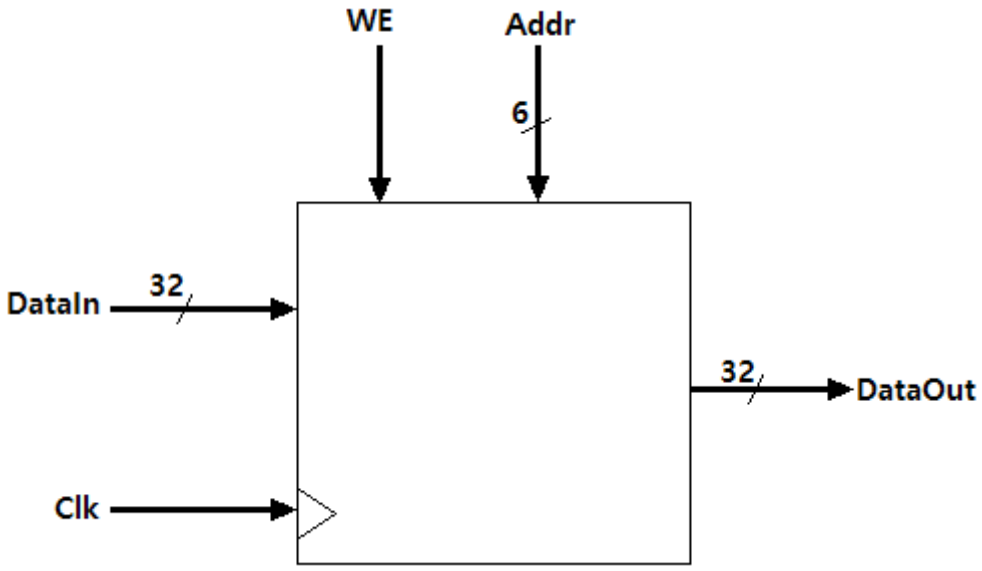
Résultat de la simulation



- Offset = 1 et nPCsel = 0. On veut l'indice 0 car PC vaut initialement 0.
- Offset = 1 et nPCsel = 1. On veut l'indice 2 car PC s'est incrémenté de 1 + (offset = 1) = 2.
- Offset = 2 et nPCsel = 0. On veut l'indice 3 car PC s'est incrémenté de 1.
- Offset = 2 et nPCsel = 1. On veut l'indice 6 car PC s'est incrémenté de 1 + (offset = 2) = 3.
- Offset = 2 et nPCsel = 1. On veut l'indice 9 car PC s'est incrémenté de 1 + (offset = 2) = 3.

nPCSel	Offset	Instruction	PC	Opération
X	**X**	**E3A01020 (MOV)**	**0**	**Valeur initiale**
1	**1**	**E6110000 (LDR)**	**2**	**PC = PC + 1 + 1**
0 (changement effectué après le front montant de l'horloge)	**1**	**E0822000 (ADDr)**	**3**	**PC = PC + 1**
1	**2**	**EBAFFFFB (BLT)**	**6**	**PC = PC + 2 + 1**
1	**2**	**EBAFFFF6 (BAL)**	**9**	**PC = PC + 2 + 1**

Mémoire de données



Entrées:

- **Clk** : L'horloge synchronisé au système, sur 1 bit
- **WE** : **W**rite **E**nable. L'écriture est autorisé lorsque le signal est 1.
- **DataIn** : Les données à écrire dans la mémoire, sur 32 bit
- **Addr** : L'adresse de la mémoire à adresse, sur 6 bit

Sortie:
- **DataOut** : Les données lues de la mémoires à l'adresse **Addr**, sur 32 bit

Fonctionnement :
La mémoire de données se fait en 64(= 2^6) registres. Donc, la taille disponible est 64 mots. Ces registres sont adressés par l'entrée d'adresse **Addr**.

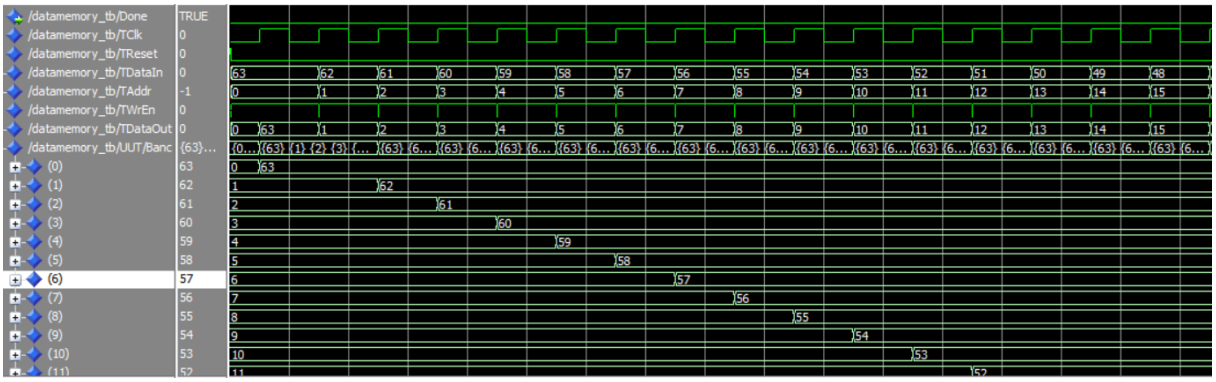
La lecture est asynchrone (combinatoire et simultanée). La sortie **DataOut** donne immédiatement la valeur actuelle du registre à l'adresse **Addr**.

L'écriture est synchrone sur le front montant de l'horloge, **Clk**. Si le signal **WE** est 1, les données passées par l'entrée **DataIn** sont écrites au front montant de l'horloge. Si le signal **WE** est 0, aucune opération d'écriture n'est permise. Donc le signal **WE** est prioritaire.

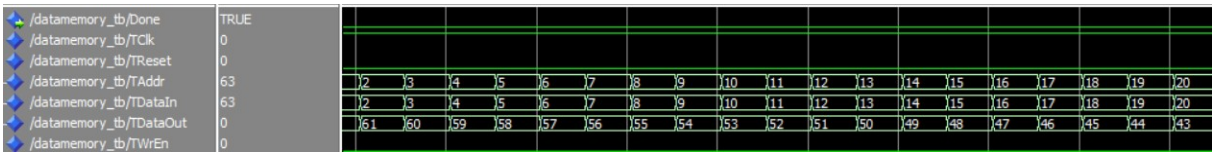
Description :
La mémoire de données a une taille plus grande que celle de la banc de registres. Les valeurs à garder sans être écrasées sont stocké ici par la commande STR. L'accès simultané à des différents registres n'est pas permis. Pour pouvoir modifier la valeur stockée, il faut charger la valeur dans la banc de registres ou écraser la valeur par une nouvelle valeur.

Banc de test

Pour valider son fonctionnement, nous avons fait une banc de tests faisant les tests suivants :
Résultat de la simulation
Un test d'écriture:

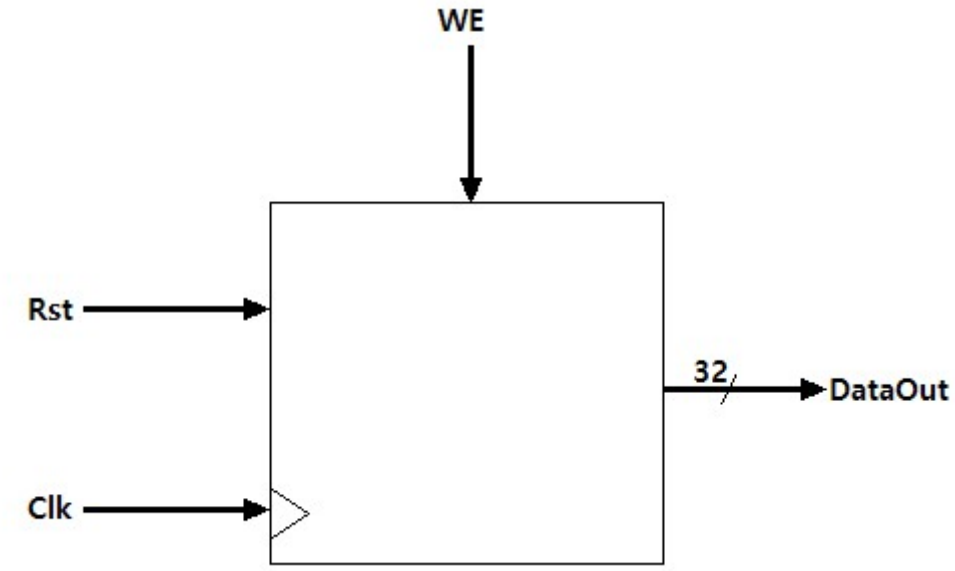


Commençant par 63, la valeur qui se décrémente de 1 est écrite dans un registre de l'adresse [0x19 - valeur]. L'écriture peut être vérifiée dans le tableau de registres **Banc.**



La lecture est effectuée de manière asynchrone et les valeurs dans la mémoire sont correctement récupérées.

Registre d'état du processeur



Entrées:

- **Clk** : L'horloge synchronisé au système, sur 1 bit
- **Rst** : Un signal Reset ou RAZ (remise à zéro). C'est actif à l'état haut et il initialise la valeur du registre à 0.
- **WE** : **W**rite **E**nable. L'écriture est autorisé lorsque le signal est 1.
- **DataIn** : Les données à écrire dans la banc des registre, sur 32 bit

Sortie:

- **DataOut** : Les données lues de la mémoires à l'adresse **Addr**, sur 32 bit

Fonctionnement :

Registre d'état du processeur est un registre de 32 bits. Ces registres sont adressés par l'entrée d'adresse **Addr**.

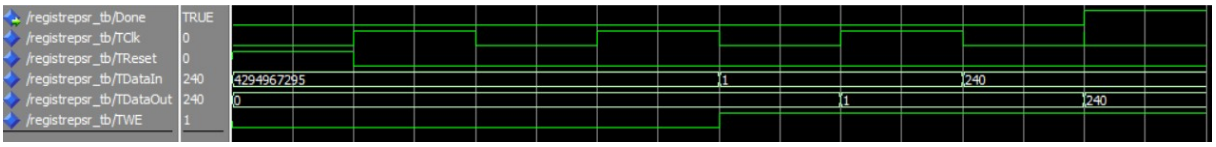
La lecture est asynchrone (combinatoire et simultanée). La sortie **DataOut** donne toujours la valeur actuelle du registre.

L'écriture est synchrone sur le front montant de l'horloge, **Clk**. Si le signal **WE** est 1, les données passées par l'entrée **DataIn** sont écrites au front montant de l'horloge. Si le signal **WE** est 0, aucune opération d'écriture n'est permise. Donc le signal **WE** est prioritaire.

Description :
Le registre d'état du processeur sert à enregistrer des drapeaux de l'unité arithmétique logique. Dans ce projet, seulement le drapeau N (négatif) est pris en compte. Le drapeau conservé est référé pour une opération arithmétique, comparaison.

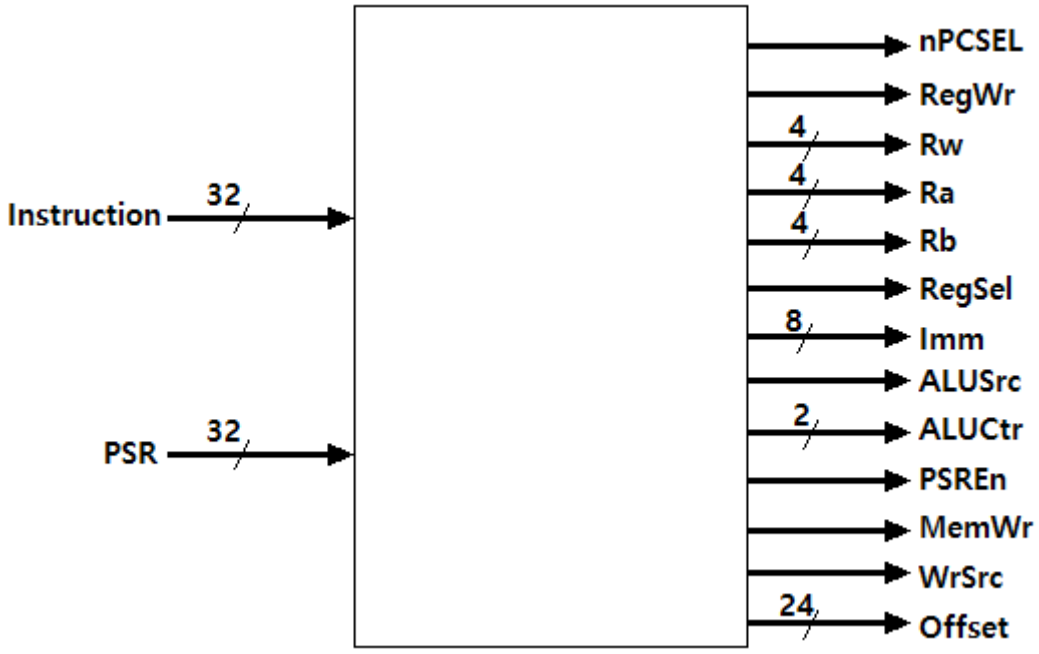
Banc de test

Pour valider son fonctionnement, nous avons fait une banc de tests faisant les tests suivants :
Résultat de la simulation



L'écriture est faite de manière synchrone.

Décodeur d'instructions



Entrées:
- **Instruction** : **W**rite **E**nable. L'écriture est autorisé lorsque le signal est 1.
- **PSR** : Les données à écrire dans la banc des registre, sur 32 bit

Sortie:

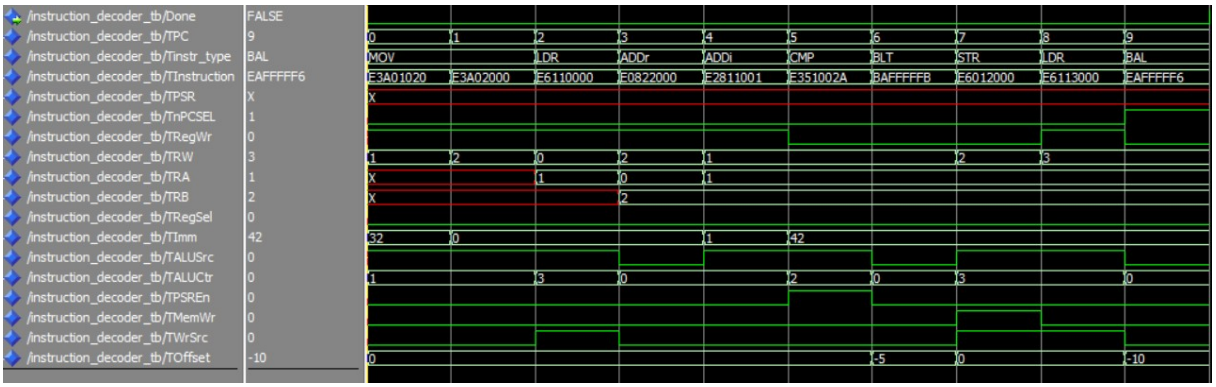
- **nPCSel** : L'entrée **nPCSEL** de l'unité de gestion des instructions, sur 1 bit.
- **RegWr** : L'entrée **WE** de la banc de registre, sur 1 bit.
- **Rw** : L'entrée **Rw** de la banc de registre, sur 1 bit.
- **Ra** : L'entrée **Ra** de la banc de registre, sur 1 bit.
- **Rb** : L'entrée **Ra** de la banc de registre, sur 1 bit.
- **RegSel** : L'entrée **RegSel** de la banc de registre, sur 1 bit.
- **Imm** : L'entrée **A** de l'extension de signe, sur 8 bit.
- **ALUSrc** : La commande appropriée pour le multiplexeur qui choisit entre la sortie de la banc (bus **B**) de registres et la sortie(**S**) de l'extension de signe , sur 1 bit.
- **ALUCtr** : L'entrée **OP** de l'unité arithmétique logique, sur 2 bit.
- **PSREn** : L'entrée **WE** du registre d'état du processeur, sur 1 bit.
- **MemWr** : L'entrée **WE** de la mémoire de données, sur 1 bit.
- **WrSrc** : La commande appropriée pour le multiplexeur qui choisit entre la sortie de l'UAL (**S**) et la sortie de la mémoire données (**DataOut**), sur 1 bit.
- **Offset** : L'entrée **Offset** de l'unité de gestion des instruction, sur 8 bit.

Fonctionnement / Description:
Le décodeur d'instructions décode une instruction en 32 bits et pour commander les multiplexeurs et passer les données appropriées à tous les unités, qui en a besoin, dans le processeurs. Alors, le processeur traite les donnée de manière désirée par des utilisateurs. Il est asynchrone.

Instruction	nPCSel	RegWr	ALUSrc	ALUCtr	PSREn	MemWr	WrSrc	RegSel
MOV	0	1	1	01	0	0	0	0
ADDr	0	1	0	00	0	0	0	0
ADDi	0	1	1	00	0	0	0	0
CMP	0	0	1	10	1	0	0	0
LDR	0	1	1	11	0	0	1	0
STR	0	0	1	11	0	1	1	0
BAL	1	0	X	XX	0	0	0	0
BLT	1	0	X	XX	0	0	0	0

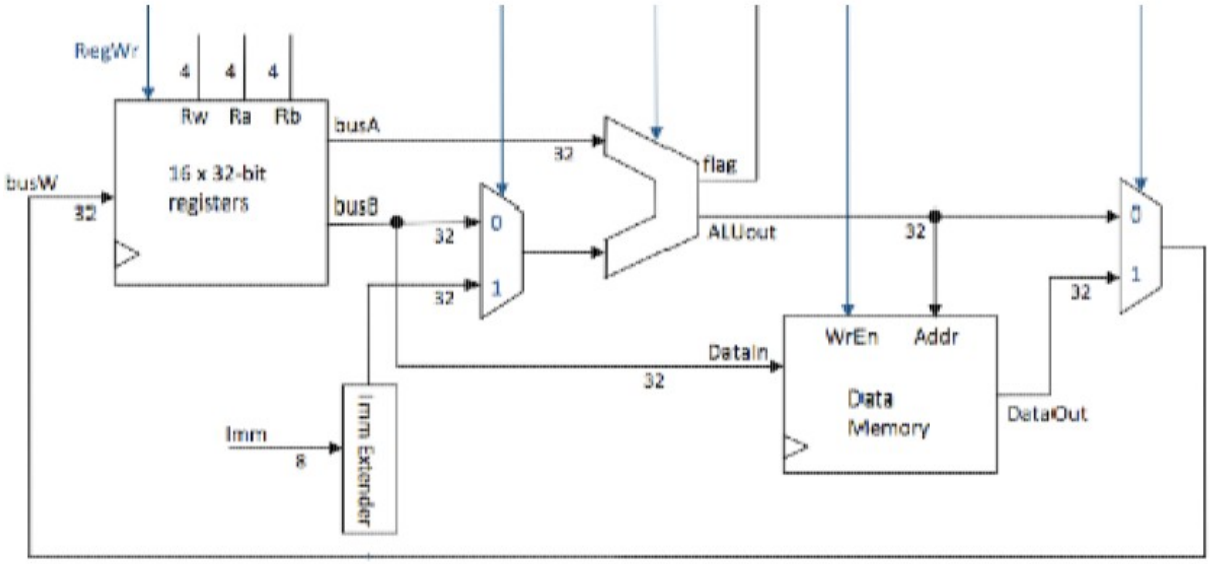
Banc de test

Pour valider son fonctionnement, nous avons fait une banc de tests faisant les tests suivants :
Résultat de la simulation



Cf. instruction_decoder_tb.vhd pour les détail du résultat.

Unité de traitement



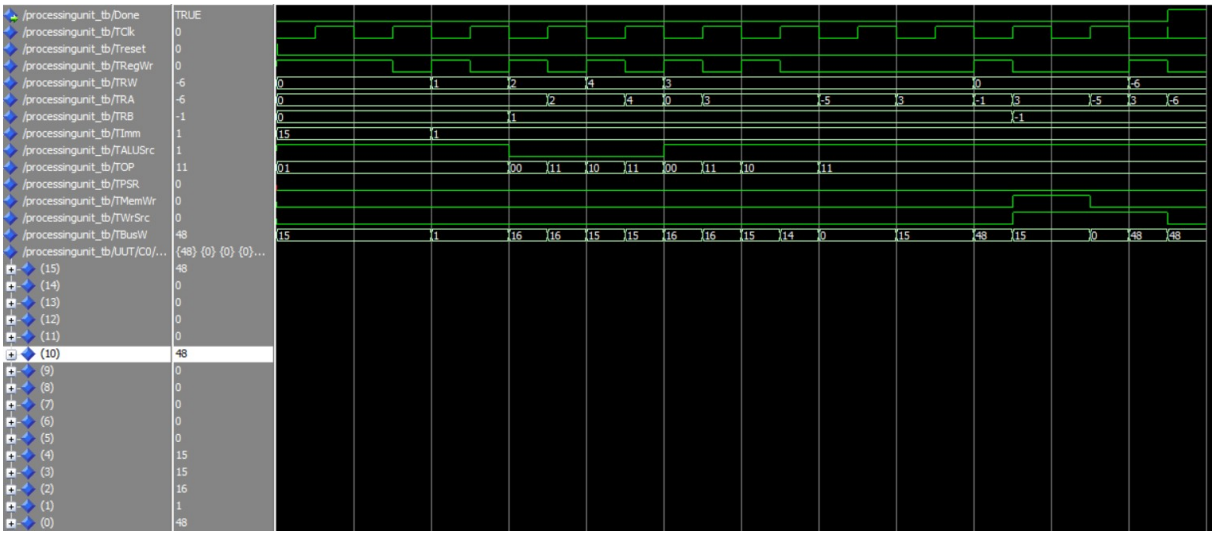
Unité de traitement est un module assemblé de:

- Banc de registre
- Unité Arithmétique Logique
- Extension de signe
- Mémoir de données
- 2 Multiplexeurs

Banc de test

Pour valider son fonctionnement, nous avons fait une banc de tests faisant les tests suivants :

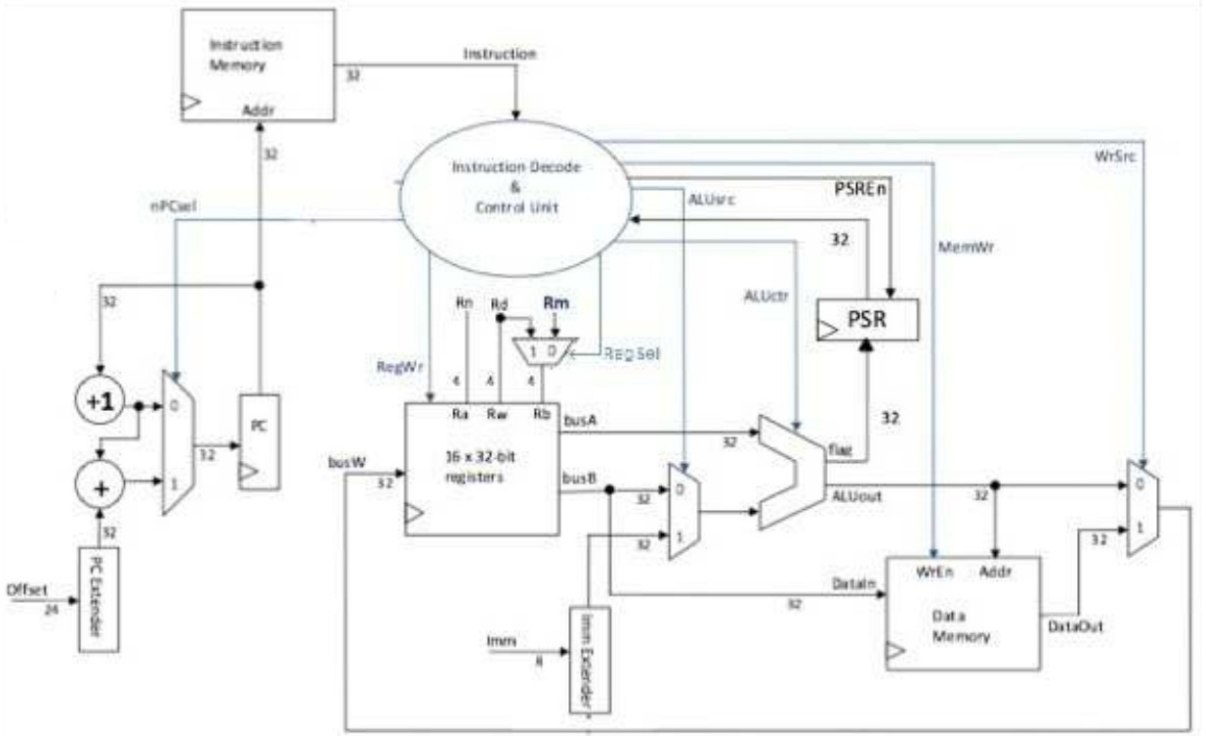
Résultat de la simulation



Cf. ProcessingUnit_tb.vhd pour les détail du résultat.

Processeur du monocycle

Assembler tous les module fait un processeur du monocycle :



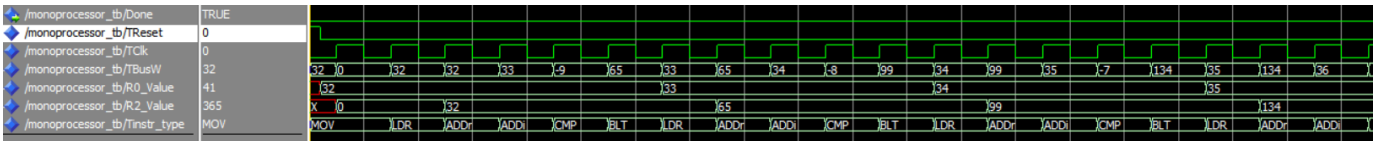
Banc de test

Pour valider son fonctionnement, nous avons fait une banc de tests suivante le program de test ci-dessous:

Programme de test

```
0x0   _main :    MOV R1,#0x20     ; --R1 <= 0x20
0x1              MOV R2,#0        ; --R2 <= 0
0x2   _loop :    LDR R0,0(R1)     ; --R0 <= DATA_MEM[R1]
0x3              ADD R2,R2,R0     ; --R2 <= R2 + R0
0x4              ADD R1,R1,#1     ; --R1 <= R1 + 1
0x5              CMP R1,0x2A      ; --? R1 = 0x2A
0x6              BLT loop         ; --branchement à _loop si
R1 inferieur a 0x2A
       _end   :
0x7              STR R2,0(R1)     ; --DATA_MEM[R1] <= R2
0x8              BAL main         ; --branchement à _main
```

Résultat de la simulation



R0_Value signifie la valeur du registre 0

R2_Value signifie la valeur du registre 2

Cf. MonoProcesseur_tb.vhd pour les détail du résultat.