PROCESSEUR MONO-CYCLE

BOUILHOL Raphaël, RENARD Valentin

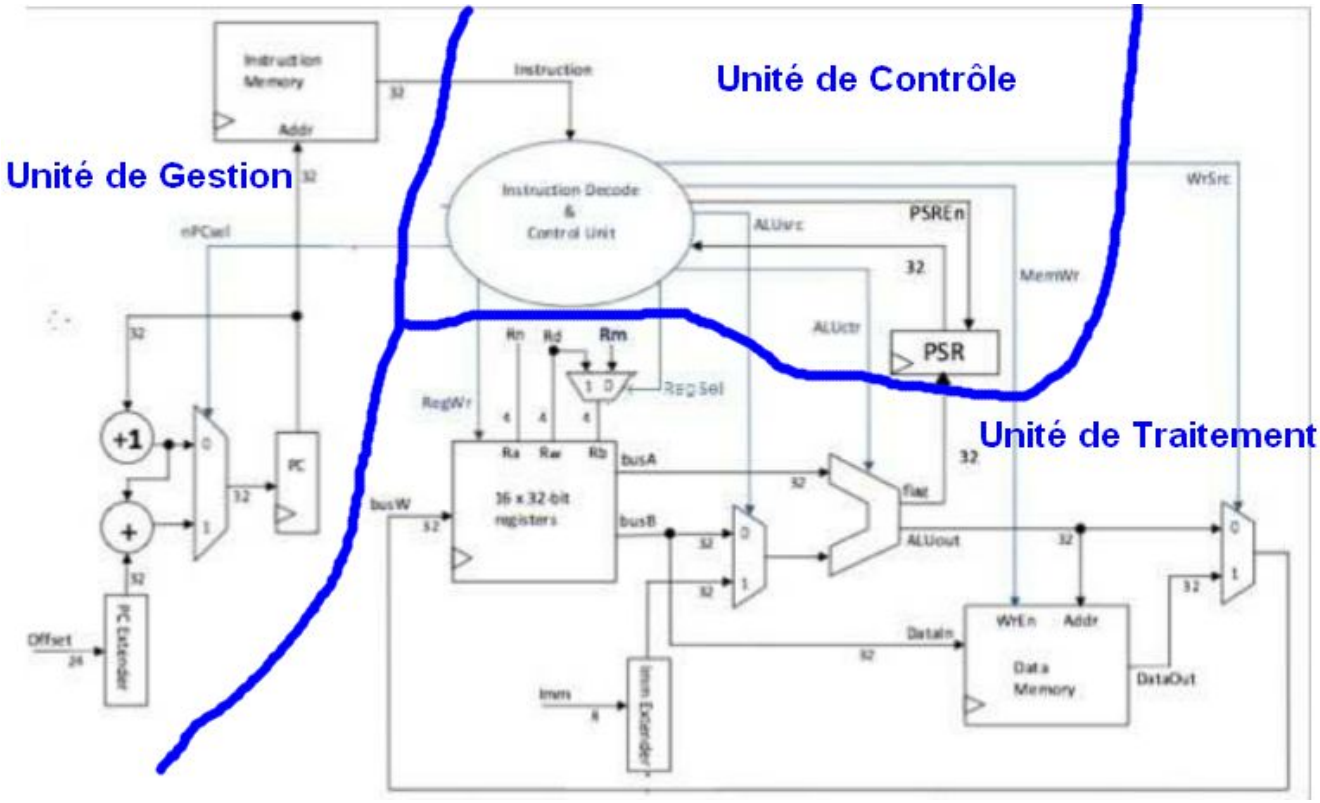


Schéma représentatif du Processeur et de sa division en unités

Le processeur est divisé en 3 unités avec des rôles différents:

- **L'unité de Gestion** qui a pour rôle de la gestion de l'instruction choisie pour être exécuté. Elle a pour cela une mémoire d'instruction contenant les lignes de code binaire à exécuter, et un registre PC contenant l'adresse de la ligne à exécuter dans la mémoire.
  Deux cas sont alors présent: soit l'instruction courante est un saut et la valeur du registre PC pourra être amené à se voir additionner la valeur de l'offset (MUX à 1), soit l'instruction n'est pas un saut et la valeur du PC sera simplement incrémenté de 1 à chaque front montant d'horloge (MUX à 0).

  <u>Composition:</u>
  - Mémoire d'instruction de 64 mots de 32 bits;
  - Registre PC, disposant d'un reset et actualisant sa valeur de sortie à chaque front montant, donnant l'adresse de l'instruction à fournir à la mémoire d'instructions;
  - Extendeur de signe faisant passer la valeur de l'Offset de 24 à 32 bits;
  - Multiplexeur gérant le choix à donnant en entrée au PC entre PC+1 et PC+Offset+1 en cas de saut.

- **L'unité de Contrôle** qui une instruction donné, contrôlera tous les signaux fournis à l'unité de traitement (AUT) ainsi que l'offset et la valeur du MUX de l'unité de contrôle.
  Selon le type d'instruction, les signaux de contrôle des composants vont changer selon le tableau suivant pour que l'instruction puisse être effectué. Les valeurs des registres Rn, Rd et Rm, de l'offset et de l'immédiat vont elles aussi être déduite du code binaire de l'instruction, et donné à l'AUT, qui va les utiliser ou non selon le type d'instruction.

  Composition:
  - Décodeur d'Instruction, pilotant les signaux de contrôle du Processeur et les registres d'adresse fournis au banc de registres, à partir du code binaire sur 32 bits de l'instruction;
  - Extendeur de signe, faisant passer la valeur du signe de la sortie de l'ALU de 1 à 32 bits.
  - PSR, conservant l'état du signe de la sortie de l'ALU.

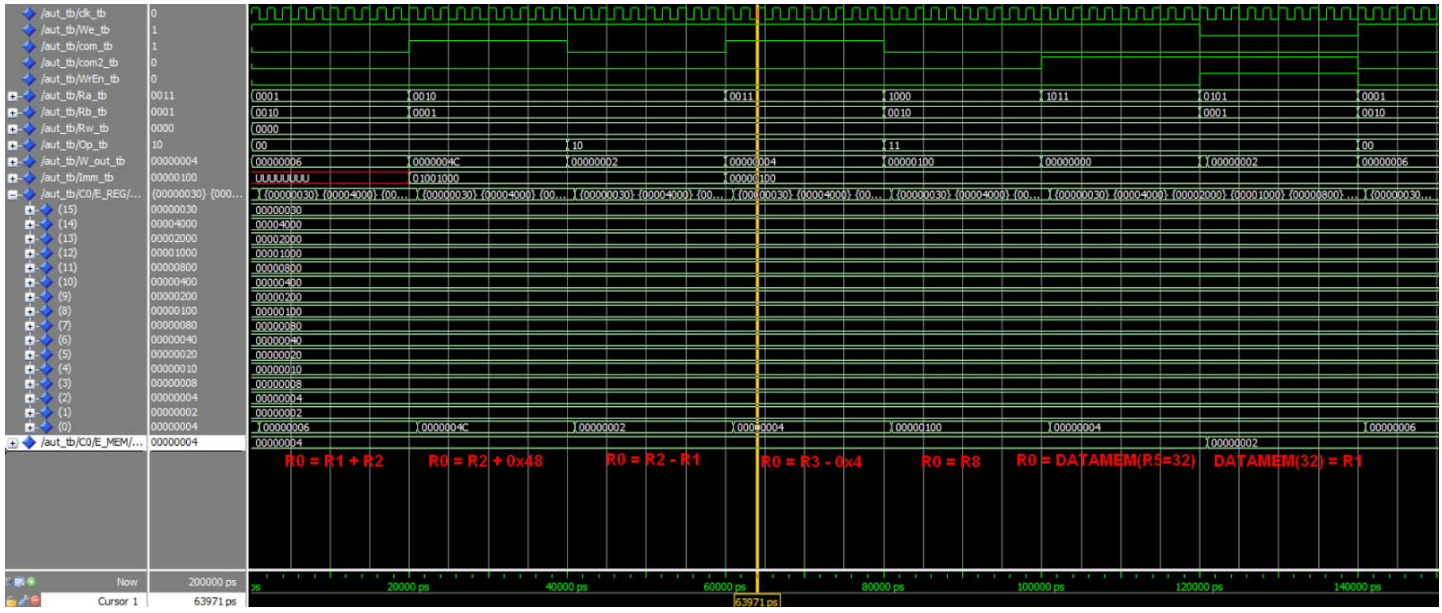
Instruction	nPCsel	RegWr	ALUsrc	ALUctr	PSREn	MemWr	WrSrc	RegSel
ADDi	0	1	1	00	1	0	0	0
ADDr	0	1	0	00	1	0	0	0
BAL	1	0	0	00	0	0	0	0
BLT	0/1	0	0	00	0	0	0	0
CMP	0	0	1	10	1	0	0	0
LDR	0	1	0	11	0	0	1	0
MOV	0	1	1	01	1	0	0	0
STR	0	0	1	00	0	1	0	1

Tableau des signaux de contrôle en fonction du type d'instruction

- **L'unité de traitement** qui va mettre en oeuvre les instructions. Il contient le banc de registres, la mémoire et l'unité arithmétique et logique (ALU), permettant d'effectuer des additions et soustractions entre registre et registre ou registre et immédiat. Des lectures et écritures du banc de registre ou de la mémoire sont également au coeur de l'unité de traitement.
  Composition:
  - Banc de 16 registres de 32 bits;
  - Multiplexeur permettant de choisir entre le bus B sortant du banc de registres et l'immédiat
  - Extendeur de signe permettant à l'Immédiat de passer de 8 à 32 bits.
  - Unité Arithmétique et Logique, possédant 2 bus en entrée et un bus de sortie contenant

le résultat de l'opération (A+B, A-B, A ou B);
- Mémoire de 64 mots de 32 bits;
- Multiplexeur permettant de choisir le bus de sortie de l'ALU ou de la Mémoire pour le bus de sortie de l'unité, revenant vers le banc de registres.

On souhaite à présent montrer le bon fonctionnement de l'AUT, par sa capacité à effectuer tous les types d'opération possible d'addition et de lecture/écriture :



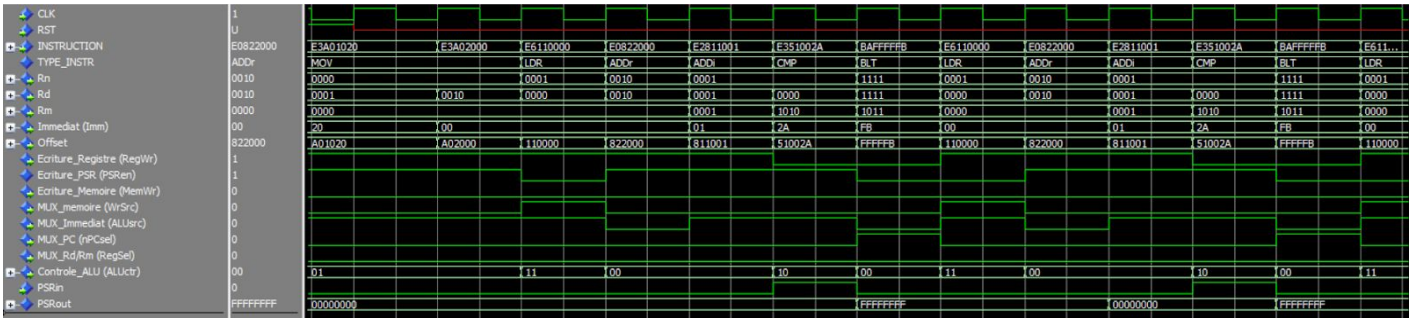
Simulation de l'AUT

Ici sont affichés **tous les signaux de contrôle de l'AUT** (**com** contrôle le multiplexeur de l'immédiat, **com2** celui à la sortie de la mémoire), **les registres du banc de registres** et le **32e registre de la mémoire**. Tous les résultats des opérations sont stockés dans le registre 0 du banc de registres. Les opérations effectuées sont affichées sur la capture d'écran.

On peut voir que l'AUT parvient à effectuer tous les types d'opérations, entre registres et registres, registres et immédiat, et registre et mémoire.
L'élément particulier est que l'écriture des registres du banc de registre ou de la mémoire se fait un coup d'horloge après que l'instruction ait été donné. Cela nous a obligé à faire les écritures non pas sur fronts montants mais sur fronts descendants. En effet, avoir les écritures avec un coup d'horloge de retard provoquait des erreurs, car les instructions sont changées à chaque front montant. Les commandes de contrôle du banc de registre et de la mémoire peuvent donc très bien changer entre temps à la réception d'une nouvelle instruction.
Mettre les écritures sur fronts descendants nous a permis de résoudre ce problème.

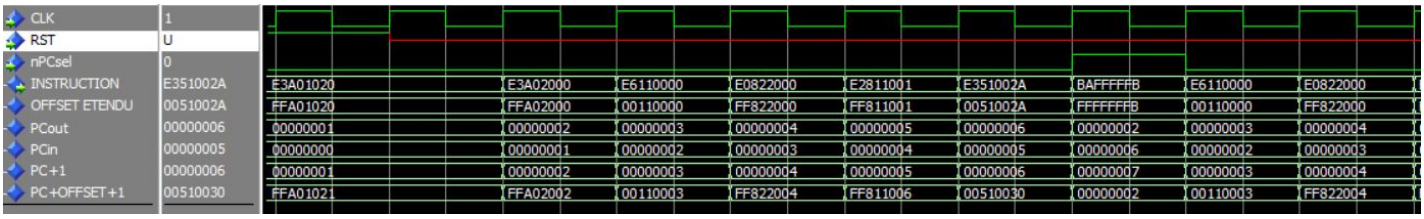
On souhaite maintenant instancier le processeur entier, à travers ces 3 unités. En simulant le code fournis et en affichant le début de son exécution, on peut observer le bon fonctionnement de chaque unités:

- **<u>Unité de contrôle</u>**



Ici sont affichés les entrées (**Instruction**, flag de l'ALU (**PSRin**), **clk et rst**) et **tous les signaux contrôles pilotés par l'unité**.
Tous les changements d'instructions induisent bien leurs changements d'associés de valeurs des registres Rn, Rd, Rm, de l'Immédiat et de l'Offset, et des signaux de contrôles si le type d'instruction change.
L'écriture du PSR est elle aussi fonctionnelle. Par exemple, lors du CMP, l'écriture est autorisé et le flag est à '1', la valeur stocké dans le PSR passe donc à 0xFFFFFFFF, et ne repasse à 0 que quand l'écriture est ré-autorisée.

- **<u>Unité de gestion d'instruction</u>**



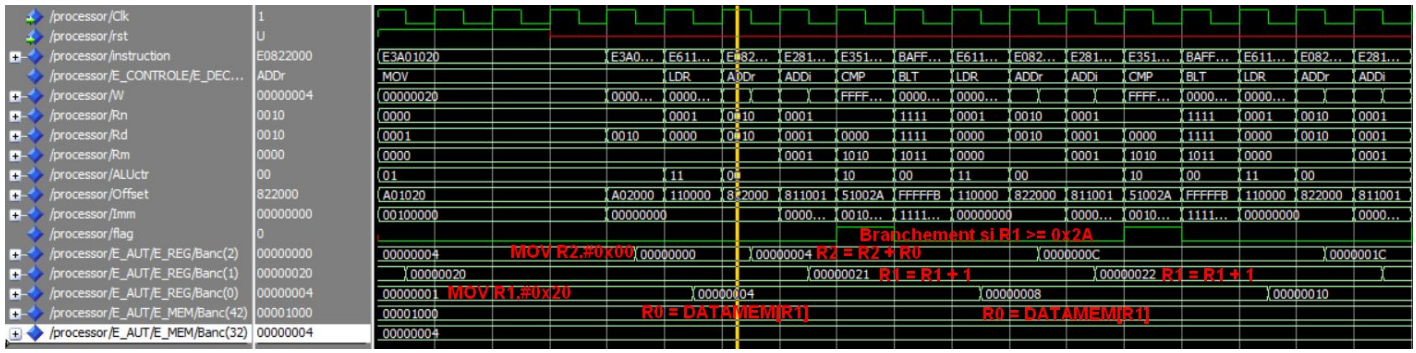
Ici sont affichés les entrées (**clk, rst** et **nPCsel** le **signal de contrôle du MUX**), la sortie **Instruction**, et les **signaux internes pilotant l'adresse de l'instruction choisie**.
On peut voir les deux valeurs possibles du PC s'actualiser correctement à chaque front montant, et que avoir le nPCsel à '1' a bien l'effet escompté, c'est à dire que le PCin prenne la valeur de PC+Offset+1 et non PC+1.

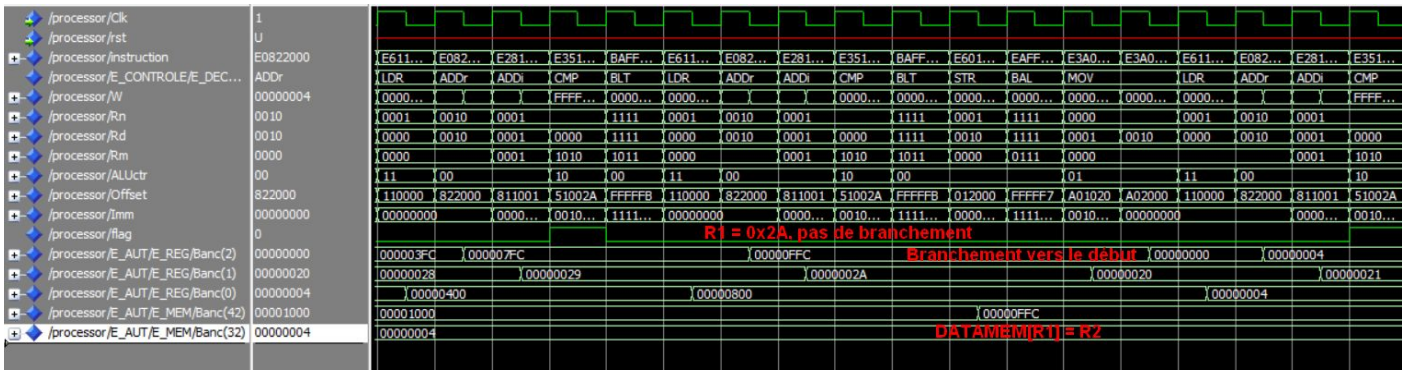
- **<u>Unité de traitement</u>**

Ici sont affiché les signaux d'entrées (**signaux de contrôle des composants, registres Ra, Rb et Rw, l'immédiat, la clock et le reset**) et les signaux internes à l'unité (**bus A, B et de sortie pour l'ALU, bus de sortie de la mémoire et bus W de sortie**).
Le bon fonctionnement des bancs de registres a déjà été montré précédemment.

On peut observer que le changement des signaux de contrôles sont bien transmis de l'unité de contrôle à l'unité de traitement, que les signaux relatifs à l'ALU prennent tous les bonnes valeurs.
Le bus W de sortie changent de valeurs au front descendant des opérations d'addition. dû à l'écriture sur front descendant du bancs de registres car le registre d'adresse Rw et Ra du banc ont la même valeur.

On veut enfin montrer une simulation globale du processeur soumis au code fournis pour comprendre son exécution. On obtient les simulations suivantes:





Simulation du processeur entier avec le code fourni

Ici sont affichés la valeur hexadécimal de l'instruction, le type d'instruction correspondant, le bus de sortie de l'AUT (**W**), les signaux **Rn**, **Rd**, et **Rm** du banc de registre, le **signal de contrôle de l'ALU**, **l'offset du PC**, la valeur de **l'immédiat**, le **flag** de sortie de l'ALU, **les registres 0, 1 et 2 du banc de registres** et **les registres 33 et 42 de la mémoire**. Les instructions effectués sont inscrites sur la première capture d'écran, à côté ou sous le registre qu'elle affecte.

Le but du code fourni est d'initialiser les valeurs des registres 1, 2, puis d'effectuer une boucle qui stocke la valeur du registre n°R1 de la mémoire dans le registre 0 du banc, incrémente R2 de la précédente valeur obtenu et enfin incrémente R1 de 1.
La valeur de R1 est ensuite comparé à celle de 0x2A. Si R1 est inférieur, le flag de sortie de l'ALU sera à 1 et un branchement 4 instructions en arrière sera effectué vers le LDR R0,0(R1).
R1 va donc s'incrémenter jusqu'à atteindre 0x2A (screen 2). Au moment du BLT, le flag n'est pas à 1, le branchement n'est pas effectué et le code continu pour faire le STR R2,0(R1), avant de revenir au tout début.

On peut voir que la valeur des registres que l'on peut déduire du code de l'instruction (Rn, Rd, Rm, Immédiat, Offset) sont bien actualisée à chaque changement d'instructions, que le bus de sortie change de valeur à chaque front d'horloge quand une addition est effectué, changement venant de l'écriture sur front descendant des registres du banc de registre et de la mémoire.

Le processeur est donc entièrement fonctionnel.