BLANCHÉ Thomas HONORÉ Alexandre SITHIDEJ Clara SYLVESTRE Antonin

Projet Processeur

3. DecodeIR

Question : Quel comportement doivent avoir les composants RegPC et GetAddr dans le cas d'un programme sans instruction mémoire ni de contrôle ?

Si aucune instruction mémoire ni de contrôle n'est utilisée dans le programme, RegPC et GetAddr ne seront utilisé que lors des Fetch.

RegPC ajoutera donc 1 à chaque Fetch a l'adresse PC pour exécuter l'instruction suivante.

GetAddr ne sera jamais utilisé lors d'un Exec, il ne fera donc que laisser passer l'adresse PC à chaque Fetch.

4. Instructions MEM

1. Quel est le rôle des composants RegPC et GetAddr ?

RegPC : ce composant va nous permettre de calculer la prochaine adresse mémoire qui doit être exécuté. Il est donc utilisé durant le Fetch. Il y a pour l'instant qu'une possibilité qui est d'ajouter 1 à AddrPC.

GetAddr : ce composant va nous permettre de choisir la prochaine adresse mémoire a la quel nous voulons accéder.

2. Pour implémenter les instructions MEM, il faut adapter le composant GetAddr. Que doit valoir sa sortie lors des phases Fetch et Exec ?

GetAddr : Si nous somme sur Fetch, ce sera l'adresse donné par RegPC (adresse de la prochaine instruction). Si nous sommes lors d'un Exec, ce sera l'adresse SR1 de l'instruction en cours de traitement. Dans le cas où c'est une instruction mémoire, l'adresse sera alors utilisée par la RAM. C'est RamCtrl qui choisira comment utiliser cette adresse. (Load, Store ou rien).

3. Proposer un encodage des instructions LD et STR. Implémenter DecodeIR en conséquence.

Cf. Traducteur.

5. Sauts

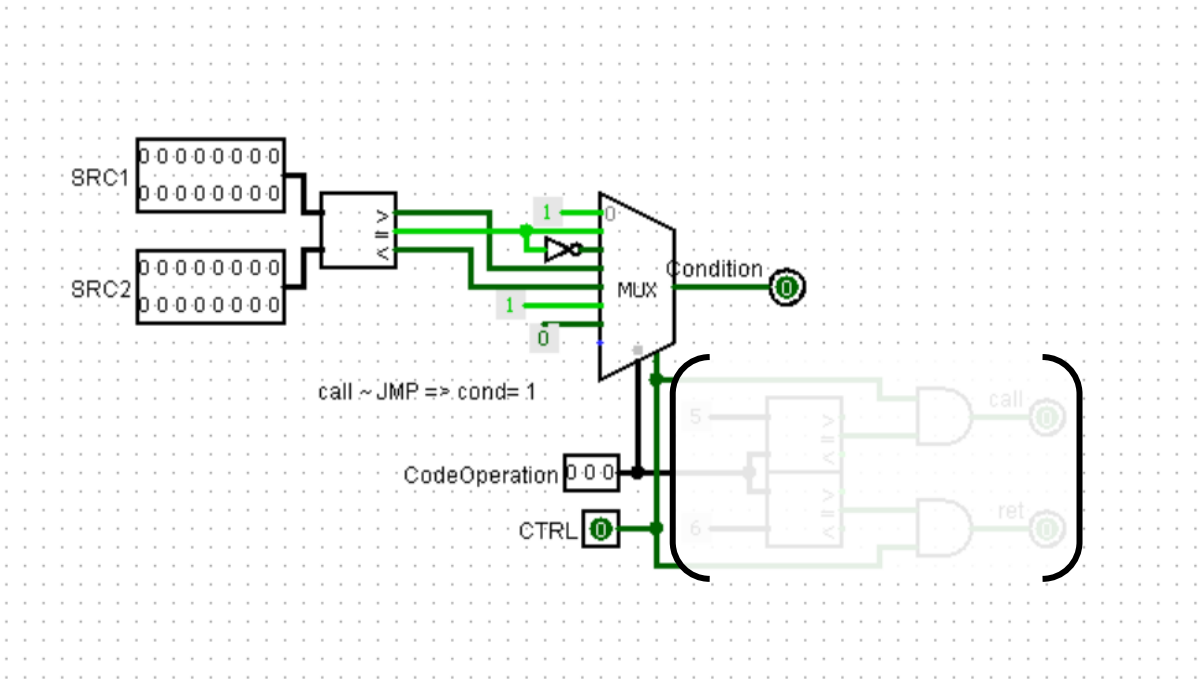
7. Choisir (intelligemment 1) un encodage et actualiser DecodeIR et votre assembleur en conséquence.

Cf. Traducteur.

8. 9. Pour commencer, on considère uniquement l'instruction JMP. Implémenter le traitement de cette instruction dans RegPC.

Gérer ensuite les instructions JEQU, JNEQ, JSUP et JINF. On pourra ajouter un sous-circuit permettant de tester les conditions recherchées.

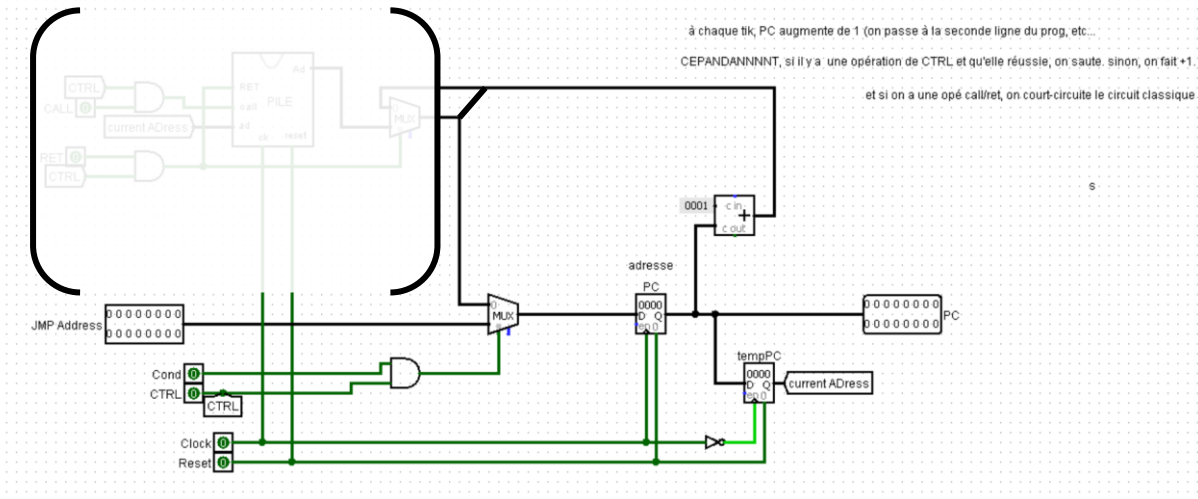
Composant (sous-circuit) CTRL : Vérifie si la condition de Jump est bien valide.



Composant RegPC :

Cond : est a 1 si la condition de jump a été vérifiée.

CTRL : est a 1 si on est bien dans une instruction de contrôle.

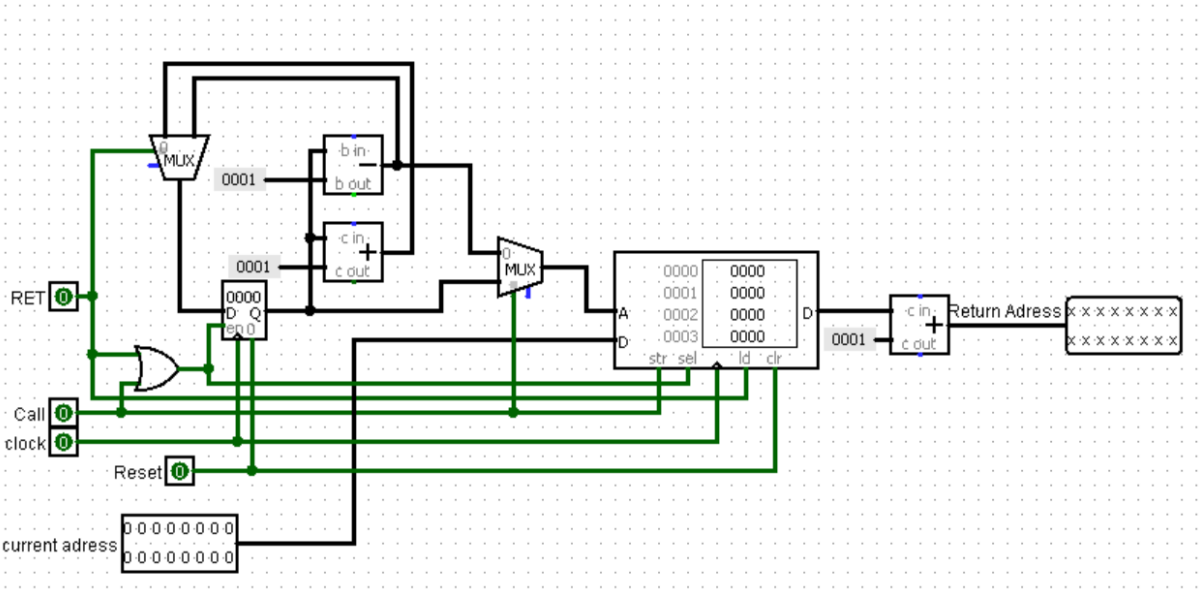


6. Appel de fonction

Il reste à implémenter les instructions CALL et RET.

11. Créer un composant Pile contenant une mémoire sur laquelle on peut empiler ou dépiler des informations. La sortie de ce composant sera toujours le contenu en haut de la pile.

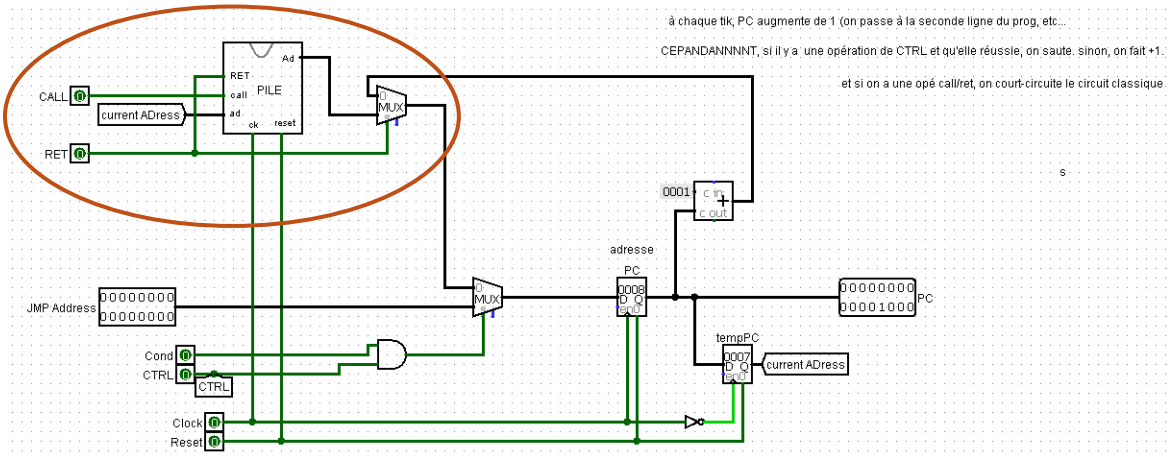
Nous avons eu quelques difficultés pour effectuer les +1 ou -1 au bon moment mais nous sommes finalement parvenus à faire cela :



Notre pile retourne la valeur du haut de la pile **+1** car lors d'un RET on revient une ligne après le CALL initial.

12. En utilisant Pile, modifier RegPC pour implémenter CALL et RET.

Nous choisissons selon si c'est un CALL ou un RET si on empile ou dépile. Dans le cas d'un RET, on le met ce qu'on dépile dans PC.



13. Tester

Nous avons testé notre programme avec un fonction qui calcule récursivement (en utilisant la pile) le factoriel d'un chiffre (Inferieur a 7 sinon cela n'est pas stockable dans un registre 16 bits...). Et cela a marché !