INF3500 – Hiver 2022

Exercices #9 – Processeur à usage général

0901 Introduction

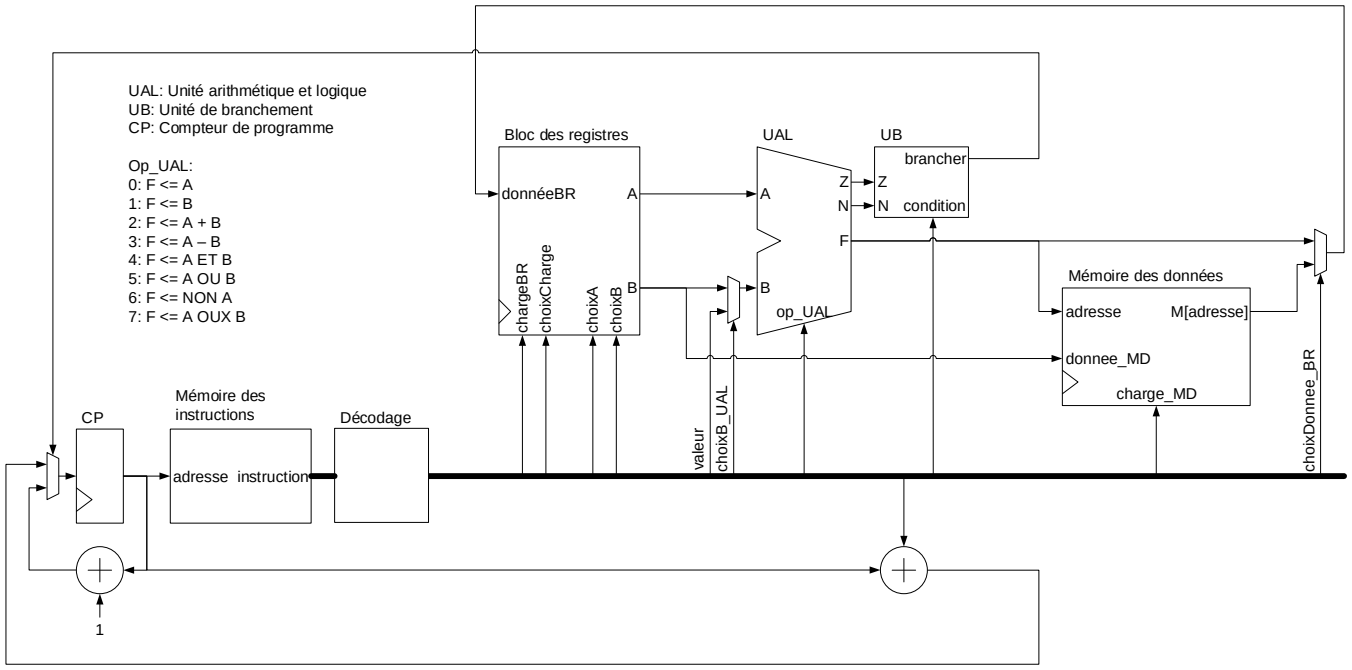
1. Comparez la complexité du processeur à usage spécifique du transmetteur RS-232, celle du joueur de black-jack, et celle du processeur à usage général Blackfin. Donnez un ordre de grandeur en termes de LUT et de bascules nécessaires pour faire l'implémentation dans chaque cas.

0902 Chemin des données

2. Considérez le chemin des données du processeur PolyRISC. Donnez la valeur des signaux de contrôle pour effectuer les opérations suivantes.

opération	chargeBR	choixCharge	choixA	choixB	valeur	ChoixB_UAL	Op_UAL	Charge_MD	choixDonnee_BR
R7 := R6 OUX R7									
R14 := 79									
M[5 + R4] := R2									
R5 := M[31 + R5]									

3. Modifiez le chemin des données du processeur PolyRISC pour permettre des instructions à trois opérandes telles que S := S + A × B, max(A, B, C) et médiane(A, B, C). Montrez quels signaux de contrôle supplémentaires seraient requis.



0903 Modélisation VHDL du chemin des données

4. Les diapositives du cours présentent le diagramme et le code VHDL d'un bloc de registres à une entrée et deux sorties. En vous en inspirant, donnez le diagramme d'un bloc de quatre registres à deux entrées et trois sorties. Il doit être possible d'écrire simultanément deux valeurs distinctes dans deux registres différents, et de lire simultanément les valeurs de trois registres sur trois ports différents A, B et C.

5. Expliquez pourquoi il est important de ne pas inclure de signal de réinitialisation dans la description VHDL d'une mémoire RAM qui doit être implémentée dans un FPGA.

0904 Jeu d'instructions d'un processeur à usage général

6. Donnez l'encodage des instructions du programme suivant.

Adresse	**Instruction**	**bits 31-30 catégorie**	**bits 29:26 détails**	**bits 25:21**	**bits 20:16**	**bits 15:5**	**bits 4:0**
0	R15 := 1234						
1	R0 := M[R15 + 0]						
2	R1 := M[R15 + 1]						
3	Si R0 > R1, goto 6						
4	R12 := R1 – R0						
5	goto 7						
6	R12 := R0 – R1						
7	M[R15 + 3] := R12						
8	goto 8						

7. Considérez l'encodage des instructions de la catégorie 0 (op. sur les registres). On voudrait supporter 256 opérations de l'UAL au lieu des 16 permises par l'encodage des détails sur les bits 29:26. Proposez une solution.

8. Pour le processeur PolyRISC décrit dans les diapositives, donnez les instructions d'un programme qui calcule la somme de 8 valeurs gardées en mémoire dans des cellules contigües à partir de l'adresse 0.

0905 Unité de contrôle

9. On veut ajouter la prise en charge des exceptions au processeur PolyRISC. Une exception peut être lancée suite à l'exécution d'une instruction. Par exemple, si l'UAL supportait la division, l'UAL générerait une exception quand le diviseur aurait la valeur 0. Lors d'une exception, le processeur devrait brancher à une adresse définie en mémoire où se situerait un code pour traiter l'exception. Énumérez et décrivez les modifications à apporter au processeur pour prendre en charge les exceptions.

Solutions

1. Complexité de processeurs. Discussion en classe.

2. Tableau des réponses :

opération	chargeBR	choixCharge	choixA	choixB	valeur	ChoixB_UAL	Op_UAL	Charge_MD	choixDon-nee_BR
R7 := R6 OUX R7	1	7	6	7	-	0	7	0	0
R14 := 79	1	14	-	-	79	1	1	0	0
M[5 + R4] := R2	0	-	4	2	5	1	2	1	-
R5 := M[31 + R5]	1	5	5	-	31	1	2	0	1

3. Instructions à trois opérandes.

Il faudrait apporter les modifications suivantes.

Le bloc des registres devrait avoir trois sorties, A, B et C. Il faudrait donc lui ajouter un port ‘choixC’ pour pouvoir aiguiller le contenu d’un des registres au port C.

L’UAL aurait besoin d’une troisième entrée, C. Il faudrait décider si on relie cette entrée directement à la sortie C du bloc des registres ou bien si on remplace l’arrangement avec un multiplexeur de la sortie B actuelle.

Il faudrait modifier le code de l’UAL pour supporter les nouvelles instructions. Le port op_UAL pourrait donc se voir élargi afin de supporter cette nouvelle gamme d’instructions. Le bloc de décodage des instructions devrait en tenir compte.

4. Réponse.

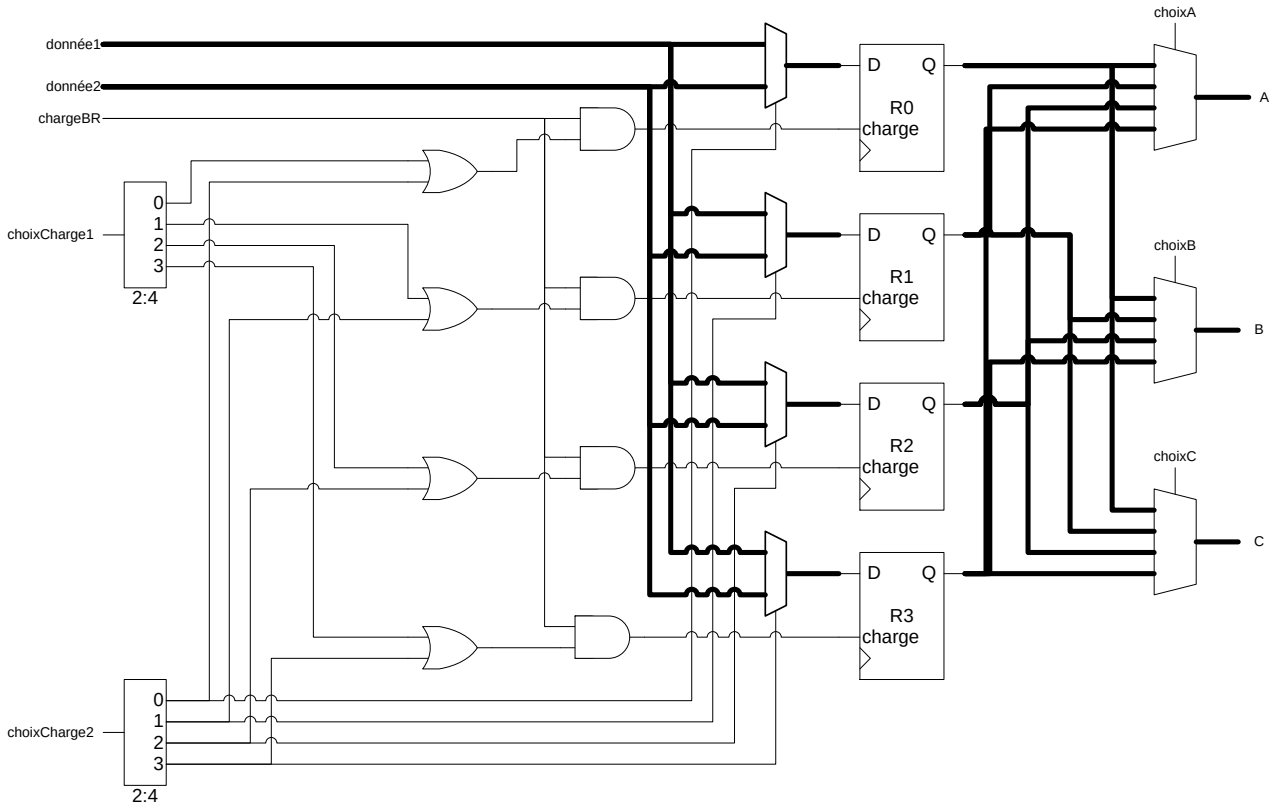
Pour la 3^{e} sortie, on instancie un 3^{e} multiplexeur.

Pour la deuxième entrée, il faut :

- Un multiplexeur à l’entrée D de chaque registre, pour choisir entre la donnée1 et la donnée2.
- Un deuxième décodeur pour déterminer quel registre doit être chargé par la donnée2.
- Une porte OU par registre.

Le fonctionnement du chargement est comme suit.

- Si chargeBR = 0, aucun registre n’est chargé.
- Si chargeBR = 1, choixCharge1 = 2 et choixCharge2 = 3, alors le signal ‘charge’ des registres 2 et 3 sera activé. Le multiplexeur à l’entrée du registre 2 recevra un 0 du 2^{e} décodeur, alors ce registre se chargera avec la valeur de donnée1. Le multiplexeur à l’entrée du registre 3 recevra un 1 du 2^{e} décodeur, alors ce registre se chargera avec la valeur de donnée2.
- Si chargeBR = 1 et que choixCharge1 = choixCharge2 = 0, alors le signal ‘charge’ du registre 0 sera activé. Le multiplexeur à son entrée recevra un 1 du 2^{e} décodeur, et alors on chargera la donnée2. Cette situation problématique est donc traitée correctement.



5. Si on inclut un signal de réinitialisation dans la description VHDL d'une mémoire RAM, le synthétiseur utilisera les bascules des tranches, ce qui gaspillera beaucoup de ressources. Il n'y a pas de signal de réinitialisation dans les blocs de mémoire distribuée.

6. Encodage d'instructions.

Adresse	**Instruction**	**bits 31-30 catégorie**	**bits 29:26 détails**	**bits 25:21**	**bits 20:16**	**bits 15:5**	**bits 4:0**
0	R15 := 1234	1	1	15	-	1234	
1	R0 := M[R15 + 0]	3	0	0	15	0	
2	R1 := M[R15 + 1]	3	0	1	15	1	
3	Si R0 > R1, goto 6	2	3	1	0	3	
4	R12 := R1 – R0	0	3	12	1	0	
5	goto 7	2	6	-	-	2	
6	R12 := R0 – R1	0	3	12	0	1	
7	M[R15 + 3] := R12	3	1	12	15	3	
8	goto 8	2	6	-	-	0	

7. On pourrait utiliser les bits 15:12, présentement inutilisés, en plus des bits 29:26, pour un total de 8 bits ce qui permettrait de spécifier 256 opérations différentes.

8. Réponse

```
       R0 := 0 -- valeur 0 pour la comparaison
       R1 := 0 -- la somme
       R2 := 7 -- indice de boucle
loop   R3 := MD[R2 + 0]
       R1 := R1 + R3
       R2 := R2 - 1
       si (R2 >= R0) goto loop
fin    si (vrai) goto fin
```

9. Prise en charge des exceptions.

Du point de vue du processeur, le traitement des exceptions est similaire à celui du branchement. Tout d'abord, on doit détecter la présence d'une exception, par exemple par l'exécution d'une instruction interdite ou dont le résultat produit un résultat à traiter par exception (comme la division par 0 ou un débordement).

Ensuite, le traitement de l'exception peut se résumer à un branchement. Le CP doit alors être chargé avec l'adresse de la première instruction d'une routine de traitement de l'exception. Cette adresse pourrait être une constante, ou, mieux, une adresse entreposée à une adresse connue – donc d'avoir un tableau de pointeurs vers des routines d'exception.

On peut aussi imaginer que le processeur effectuera une sauvegarde de son état en mémoire à partir d'une adresse déterminée d'avance. Cet état pourrait inclure entre autres les valeurs entreposées dans le bloc des registres. Il faudrait donc effectuer une longue séquence d'instructions et il semblerait préférable de programmer cette séquence.