POLYTECH DIJON

TRAVAUX DIRIGÉS

Programmation Orientée Objets / C++

Interface

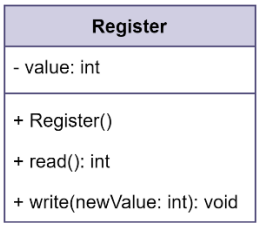
OBJECTIFS

A travers cet exercice, vous manipulerez les notions ci-dessous en C++ au cours de la réalisation d'un processeur simplifié à l'extrême :

- Classes et interfaces,
- Implémentation d'interface

REGISTER

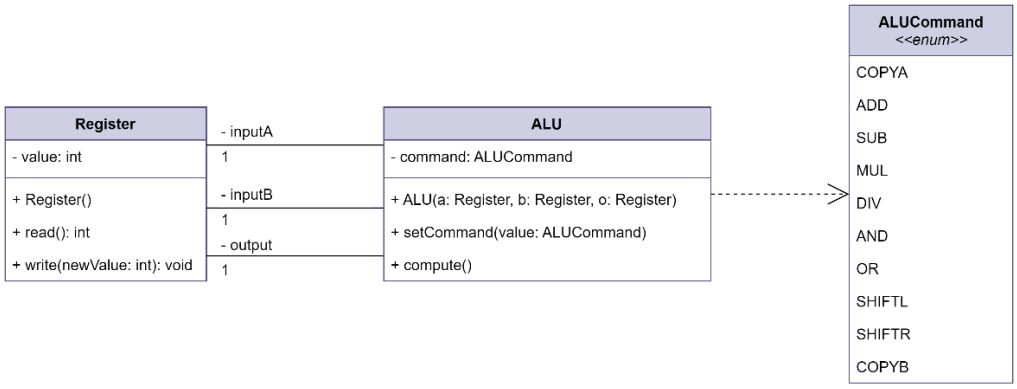
La base de tout, en informatique, est de pouvoir mémoriser une information. Le registre est l'élément du processeur qui s'occupe de cela.



- Implémentez la classe **`Register`** ci-dessus.
- Testez son bon fonctionnement dans la fonction main de votre projet.

ALU (ARITHMETIC AND LOGICAL UNIT)

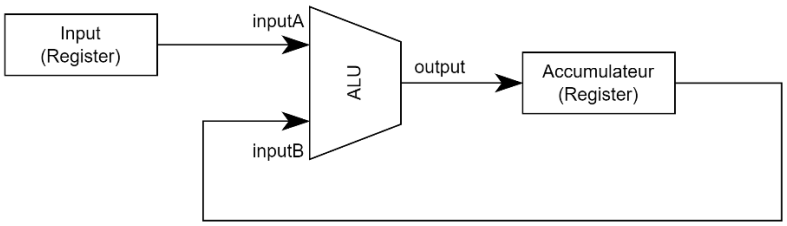
L'unité arithmétique et logique est le second élément essentiel à tout processeur qui se respecte. Un processeur n'est ni plus, ni moins qu'une calculette qui calcule très vite. L'ALU va se charger des calculs à réaliser.



- Implémentez la classe **ALU** ci-dessus en tenant compte du comportement des instructions comme suit :

COPYA	output = inputA
ADD	output = inputA + inputB
SUB	output = inputB – inputA
MUL	output = inputA x inputB
DIV	output = inputB / inputA
AND	output = inputA ET inputB
OR	output = inputA OU inputB
SHIFTL	output = inputB décalé à gauche de inputA
SHIFTR	output = inputB décalé à droite de inputA
COPYB	output = inputB

- Dans la main, « câblez » votre ALU de la façon suivante :



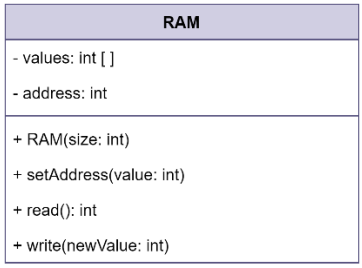
- Avec ce système, réalisez le calcul **5 + 2 – 3** et affichez la valeur contenue dans l'accumulateur à la fin des opérations.

RAM (RANDOM ACCESS MEMORY)

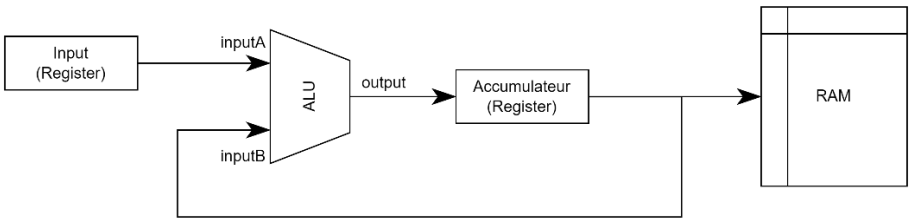
Bravo ! Vous venez de réaliser la base d'un microprocesseur.

Il pourrait être intéressant, à présent, de pouvoir conserver les valeurs de résultats précédents. Nous allons donc ajouter une mémoire de plus grande capacité qu'un simple registre : une

RAM. La RAM à l'intérêt de proposer plusieurs cellules mémoires auxquelles ont peut accéder directement via une adresse et dont on peut modifier le contenu.



- Implémentez la classe **`RAM`** ci-dessus.
- Modifiez votre « câblage » pour obtenir quelque chose comme ceci :



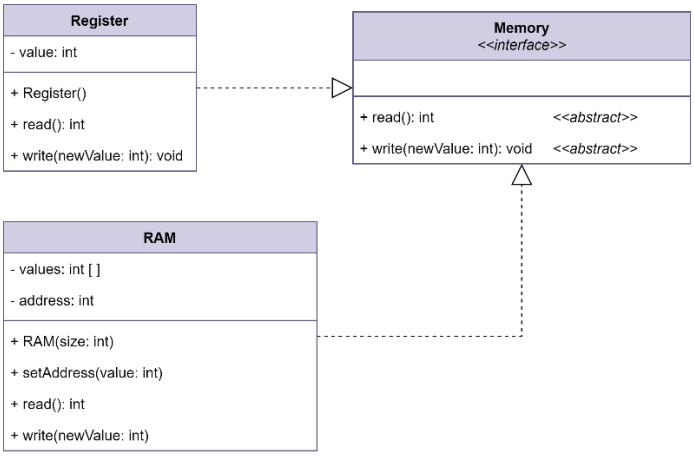
- Stockez dans la RAM le résultat de l'opération **`5 + 2 - 3`** à la fin du programme et affichez le contenu de votre mémoire.

MEMORY

Ce qui pourrait être encore plus intéressant, c'est de pouvoir connecter la RAM sur l'entrée A de l'ALU. Le problème est que cette dernière attend un registre.

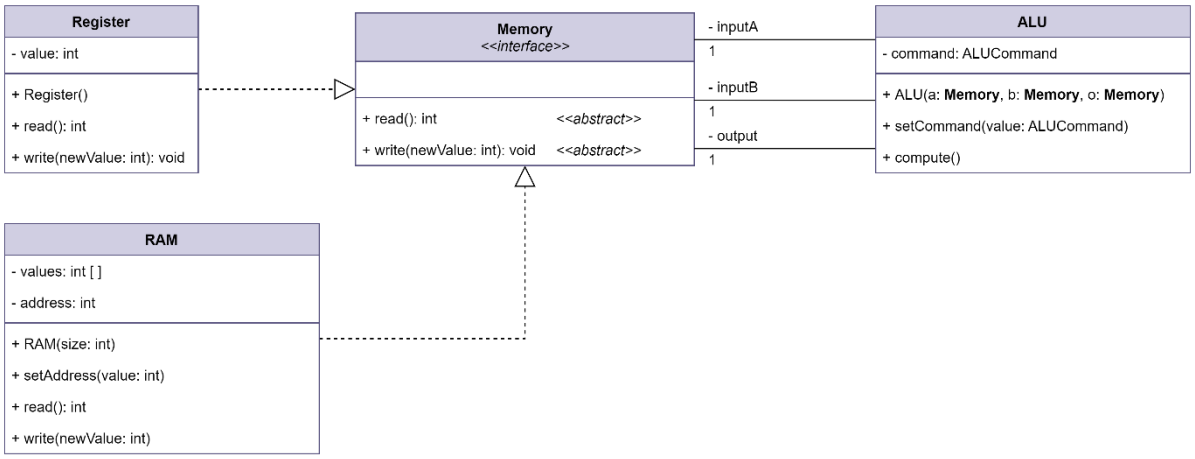
Mais si on regarde de plus près, on s'aperçoit que l'ALU n'utilise que les méthode **`read`** et **`write`** de Register. Or la RAM possède aussi des méthodes **`read`** et **`write`**.

Peut-être qu'en faisant travailler l'ALU avec quelque chose de plus abstrait qu'un registre, nous pourrions simplifier notre problème. Et c'est là qu'entre en jeu l'interface.

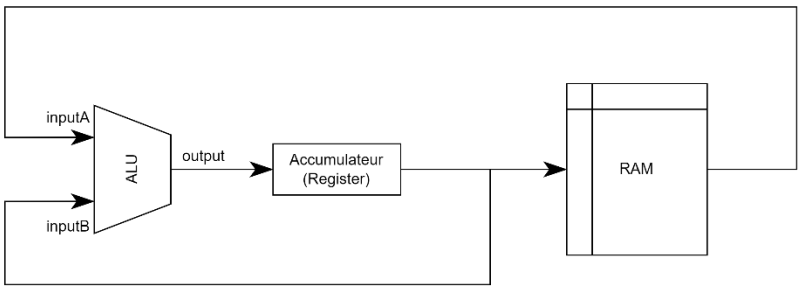


- Commencez par implémenter l'interface **`Memory`**. Il s'agit d'une simple classe dont toutes les méthodes sont virtuelle pures.
- Puis faites en sorte que **`Register`** et **`RAM`** implémente cette interface. En C++, la relation d'implémentation se traduit par une relation d'héritage.
- Sans apporter d'autre modification, vérifiez que votre programme fonctionne toujours.

- A présent, modifiez la classe **`ALU`** pour qu'elle fonctionne avec des **`Memory`** plutôt que des **`Register`** :



- Enfin, modifiez le « câblage » de votre système comme suit :

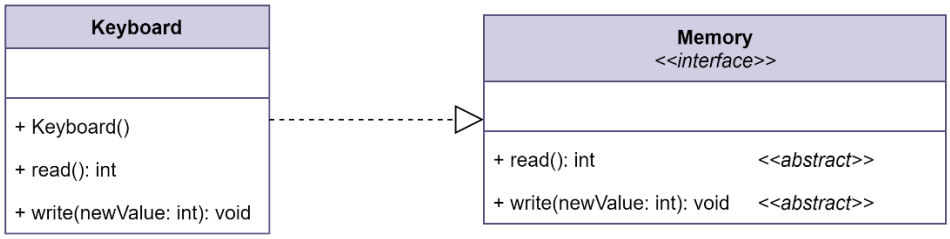


- Et réalisez le calcul suivant : **`(5 + 8) - (3 + 2)`**

> **Note**
> Vous commencerez par placer les constantes 5, 8, 3 et 2 dans la RAM

KEYBOARD

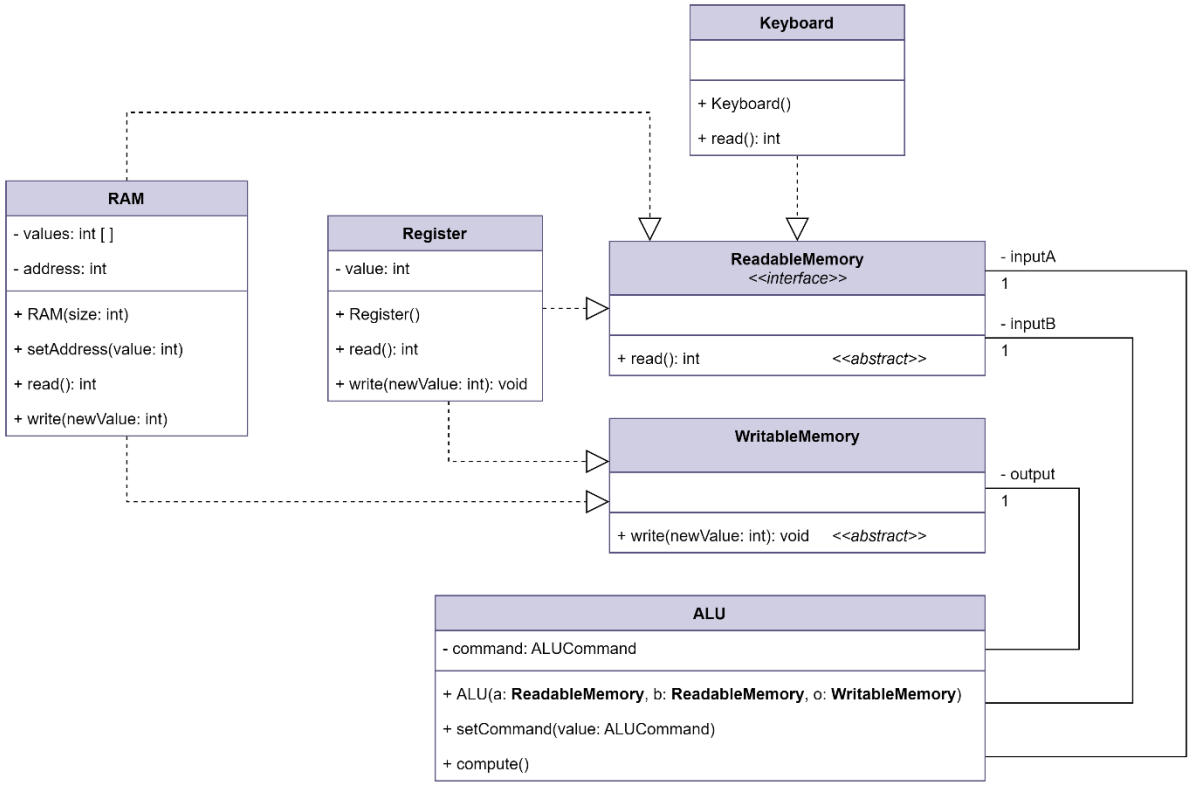
Ajoutons un peut d'interactivité à notre système.



➢ Commencez par implémentez la classe **`Keyboard`** ci-dessus.

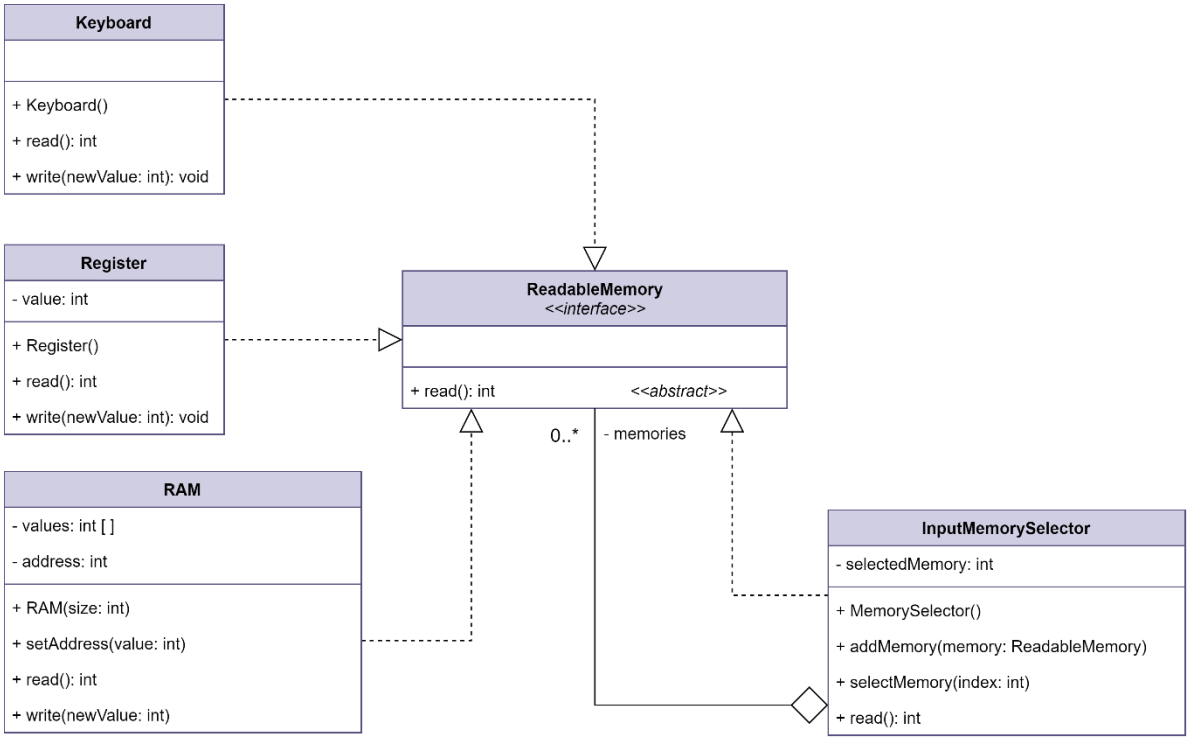
Rencontrez-vous un problème avec la méthode **`write`** de **`Keyboard`** ? C'est normal. Autant il est logique de chercher à lire une entrée clavier, autant écrire quelque chose sur le clavier de l'utilisateur n'a pas de sens. Notre modélisation souffre d'une lacune : nos mémoires sont forcément en lecture/écriture. Or, le clavier, par exemple, est en lecture seule.

➢ Modifiez le code de votre application pour répondre à la nouvelle modélisation ci-dessous :

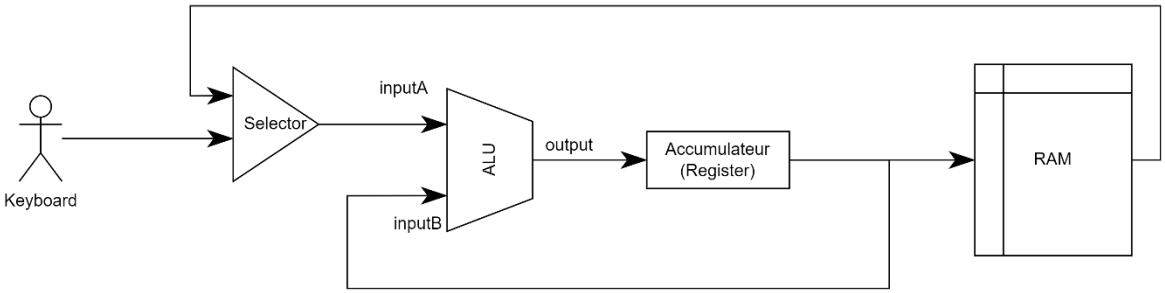


INPUTMEMORYSELECTOR

Pour terminer, il faut que l'on puisse sélectionner la source des données que l'on souhaite en entrée de l'ALU.



- Implémentez la classe **`InputMemorySelector`** présentée ci-dessus.
- Modifiez votre « câblage » pour obtenir le système suivant :



- Testez le fonctionnement avec la réalisation du calcul suivant : **`(5 + x) - (2 + y)`** où **`x`** et **`y`** seront deux valeurs saisies par l'utilisateur.