UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 1 INFO
INFO0204

TP n° 1 : Découverte ; premiers programmes

Commencez par faire un tour à la FAQ :

- pré-requis pour Info0204 : avoir Java installé ; télécharger l'archive du simulateur
- commandes utiles dans le terminal

Pour les TP d'Info0204, nous vous laissons le choix du système d'exploitation sur lequel vous travaillerez [1]. De même il vous appartient de choisir votre éditeur de texte [éditeur de texte brut] [2].

1 Prise en main du simulateur

Votre répertoire `Info0204` contient le code du simulateur de notre langage d'assemblage, sous la forme de l'archive java `archi.jar`.

Pour cette découverte du simulateur, vous aurez besoin de consulter la FAQ (*post* "Utilisation du simulateur").

1. *Chemins :*
   - Ouvrez un terminal et positionnez-vous dans le répertoire `Info0204`. Lancez le simulateur sans préciser de fichier de code assembleur et observez le résultat.
   - Déplacez vous à un autre endroit de votre arborescence (par exemple dans votre répertoire concernant `Info0203`) et lancez le simulateur depuis cet endroit.
   - A cet endroit, créez un fichier texte `pgm0.asm` [3]. Dans ce fichier, écrivez un code assembleur permettant de saisir une valeur au clavier, de la multiplier par 2 et d'afficher le résultat ; sauvegardez.
     Testez votre code (exécutez le simulateur sur ce fichier texte), sans paramètre.
   - Depuis l'endroit où vous êtes, créez un répertoire `TP1` dans votre répertoire `Info0204` et déplacez-y le fichier `pgm0.asm`.
     Testez à nouveau votre programme [complètement à distance !].
     Revenez dans le répertoire `Info0204`. Exécutez votre programme.

   Attention : Si vous avez dû fermer le fichier pour pouvoir le déplacer, rouvrez-le (*rouvrez l'éditeur sur le fichier déplacé*).
   Si vous n'avez pas eu besoin de fermer le fichier dans l'éditeur (*de fermer l'onglet affichant le contenu du fichier*), le fichier édité n'est pas le bon : si vous l'enregistrez ça ne sera pas au bon endroit. Il faut donc que vous ouvriez dans l'éditeur le BON fichier.

1. Privilégiez Linux si vous avez déjà un peu d'expérience de travail sur ce système.
2. Sous Windows, vous pouvez par exemple utiliser *Notepad* ou *Notepad++* ; sous Linux, essayez *Gedit* comme *Atom* ou *Nano* ; il existe aussi des éditeurs disponibles sur plusieurs plateformes : *Sublime Text*, *Kinsta*, *TextMate*, ... Dans le contexte d'Info0204 utiliser un IDE n'a aucun intérêt, au contraire.
3. Par convention, nos fichiers contenant des codes assembleur auront l'extension `asm` ; ceux contenant du code machine porteront l'extension `hex` s'ils sont écrits en hexadecimal, et l'extension `bin` s'ils sont écrits en binaire.

2. *Les différentes options d'exécution :*
   - Enregistrez le fichier sous le nom `pgm1.asm` ; ajoutez-y, aux deux premières lignes, vos nom et prénom, ainsi que la date ; sauvegardez ; testez votre code.
     Pour corriger l'erreur rencontrée il suffit de faire en sorte que les informations ajoutées soient des commentaires. Pour ajouter un commentaire [en fin de ligne] il suffit de le faire précéder par le caractère ';' [4].
     Une fois le code corrigé, testez-le à nouveau [sans option] : vous devriez constater que les lignes ne contenant pas de code, ainsi que les commentaires, sont ignorés dans l'affichage.
   - Exécutez votre code avec l'option `exec`, puis avec l'option `trace`.
   - L'option `pas` permet de dérouler l'exécution pas à pas, et même de modifier les éléments de votre code à chaque étape : contenu d'une case mémoire, et même de l'accumulateur ou des *flags*, voire du compteur ordinal. Cette option pourra vous être particulièrement utile pour *debugger* [5] vos codes : les dérouler jusqu'à un certain point puis essayer différentes modifications pour obtenir un certain comportement.
     Essayez ces différentes possibilités sur le code de `pgm1.asm`.
3. *Fixer des valeurs aux cases mémoires :* [`echange.asm`]
   - Ecrivez un programme permettant d'échanger les contenus des cases mémoire n° 1 et 2 de la mémoire de données. Essayez-le avec les options `exec` ou `trace`. Pour voir les choses, ajoutez des instructions pour permettre d'afficher les contenus des deux cases mémoire, au début et à la fin du programme.
   - Avec l'option `pas`, vous pouvez fixer les valeurs des cases mémoire concernées au début de l'exécution. Faites des essais.
   - La solution pour fixer des valeurs en utilisant les modes `exec` ou `trace` est d'ajouter des informations à la fin de la ligne de commande de lancement du simulateur : testez votre code en utilisant cette possibilité.

2 Premiers programmes

Les premiers exercices sont sans conditionnelles ni boucles. Ils vous permettent de tester les éléments de base de notre langage d'assemblage, et de continuer de vous familiariser avec les différentes options du simulateur.

1. Ecrivez un premier programme permettant de saisir une valeur au clavier et d'afficher le quotient de la division de cet entier par 5.
2. ... saisir une valeur au clavier et afficher son carré.
3. ... et afficher le reste de la division de 50 par cet entier (supposé non nul).

4. Ecrivez un programme permettant de consulter les contenus des cases n° 1 et 2 de la mémoire, et d'écrire dans les cases n° 3 et 4 la valeur de la somme et du produit de ces deux valeurs.
5. [`polynome.asm`] Ecrivez un programme permettant de saisir au clavier trois coefficients a, b et c ainsi qu'une valeur x, et de calculer la valeur de $P(x) = ax^2 + bx + c$.

6. Testez les codes que vous avez écrits pour les exercices de la première partie de l'énoncé de TD.

4. Il est possible d'insérer des lignes vides ou ne contenant qu'un commentaire. C'est même très utile pour structurer votre code proprement.
5. en bon français, *déboguer*

3 Premières conditionnelles

Pour permettre les branchements correspondant aux conditionnelles [et boucles] il faut numéroter les instructions.
Attention : les lignes sans code (vides ou ne contenant qu'un commentaire) ne sont pas prises en compte. Au besoin, vérifiez en utilisant le simulateur sans option sur votre programme.

1. Ecrivez un programme qui demande à l'utilisateur de saisir deux nombres a et b puis qui affiche à l'écran 1 si a est strictement supérieur b ; 0 sinon.
2. Même chose, mais cette fois-ci le programme affiche 1 si a est strictement inférieur à b (0 sinon).
3. Même chose, mais cette fois-ci pour afficher 1 *ssi* a est supérieur ou égal b.

4. Ecrivez un programme qui considère les valeurs des cases mémoire n° 1 et 2 de la mémoire de données et qui, en fonction de la parité d'une valeur saisie au clavier, y stocke soit le somme et le produit des deux premières valeurs (dans le cas où la valeur saisie est paire), soit la somme et la différence de leurs carrés (dans le cas contraire).

5. [`ord2.asm`] Ecrivez un programme permettant d'ordonner les contenus des cases n° 0 et 1 de la mémoire de données (faire en sorte qu'elles soient ordonnées, par exemple par ordre croissant). RAS si les valeurs sont 3 et 5 ; si c'est 5 et 3 on échange
6. Pour pousser plus loin le traitement vous pouvez écrire un programme permettant d'ordonner les contenus des trois premières cases de la mémoire de données.

4 Nommage des fichiers / de l'archive

Si vous souhaitez avoir un retour sur votre travail :

- nommez vos fichiers tel qu'indiqué dans l'énoncé
- renommez votre répertoire avec vos nom et prénom (sans espace ni accent)
  $\Longrightarrow$ `TP1_JAILLET_Christophe`
- si vous avez travaillé à 2 ou plus nommez le répertoire avec tous vos noms
  $\Longrightarrow$ `TP1_DEBARLE_Noel_JAILLET_Christophe_MOUSE_Mickey`
- compressez l'archive au format `zip`