Projet d’architecture des ordinateurs

Documentation pour l’utilisateur

11.02.2025

Samuel Nonon
Joshua Lozano
TD4

Projet donné par Monsieur Emmanuel Lazard

Manuel d'utilisation

Dans ce document nous allons décrire l'utilisation de notre projet d'architecture des ordinateurs, une simulation d'une machine à pile. **La taille de la mémoire de la machine à pile est de 5000.** Le répertoire Machine_a_pile est constitué, dans le dossier "scr", des fichiers :

- recup_archi.c,
- instruction.c,
- operation.c,
- read.c,
- programme.c,

de leurs headers respectifs dans le dossier "headers", d'un fichier main et d'un exécutable "simulateur.exe".

Comment construire son fichier texte?

Pour construire votre fichier texte, vous devez avoir une instruction et son argument par ligne, par exemple pour l'instruction read cela donne : read 1000, avec 1000 l'argument. Si vous écrivez read1000, cela ne marchera et l'instruction ne sera pas reconnue cependant, vous pouvez mettre autant d'espace entre l'instruction et l'argument.

Pour ajouter une étiquette vous devez le faire comme ceci : « étiquette: instruction argument ». L'espace entre l'instruction et les deux points n'est pas obligatoire, vous pouvez les coller ainsi que mettre autant d'espace que vous voulez entre les deux. L'indentation n'est pas importante, elle n'est pas prise en compte: **ici: read 1000** et **ici: read 1000** sont
write 1000 **write 1000**
lus de la même manière.
Voici un exemple extrait d'un fichier texte d'un programme renvoyant l'opposé d'un nombre donné :

```
ici: read 1000
   push 1000
   push# 0
   op 0
   jnz fin
   push 1000
   op 15
   pop 1000
   write 1000
   jmp ici
fin: halt
```

Attention : le fichier texte doit impérativement être dans créé/déplacé dans le répertoire Machine_a_pile.

Comment exécuter votre programme ?

Pour pouvoir exécuter votre programme, il vous suffira de créer/déplacer votre fichier texte dans le répertoire **Machine_a_pile** et d'écrire dans le terminal **./simulateur.exe nom_fichier.txt.**

Erreurs à ne pas faire

Le programme est conçu pour exécuter le programme dans votre fichier pgm.txt, cependant si des erreurs sont présentes dans votre fichier, le programme ne s'exécutera pas et vous renverra le type de l'erreur ainsi que la ligne à laquelle elle se situe. Voici une liste exhaustive des erreurs qui peuvent arrêter l'exécution de votre programme :

- Les fautes de frappes dans vos instructions, arguments ou étiquettes
- Ne pas donner d'argument à une instruction qui en attend un ou en donner un à une instruction qui n'en attend pas
- Ne pas mettre **exactement un espace** entre l'instruction et l'argument
- Entrer un mauvais argument (toutes les fonctions demande un entier positif sauf push# qui peut prend aussi les négatifs)
- Mettre un argument qui n'est pas dans [0;15] dans op
- Définir plusieurs étiquettes ayant le même nom
- Définir une étiquette qui commence par un chiffre
- Ne pas donner de nom à votre étiquette (écrire par exemple ":")
- La taille de la mémoire est de 5000, toute instruction utilisant des accès à certaines adresses ne peut pas avoir un argument **supérieur à 4999 ou négatif**.
- Les erreurs mathématiques comme la division par 0.

Erreur que vous pouvez faire

Voici une liste "d'erreurs" que le programme ne prend pas en compte et qui ne gênera pas l'exécution de votre programme:

- Ne pas indenter correctement votre programme
- Mettre des accents, caractères spéciaux dans le nom de vos étiquettes
- Faire un saut de ligne
- Entrer un entier supérieur à 32767 ou inférieur à -32768

Résultat

Lorsque vous exécutez votre programme, si une erreur de syntaxe est détectée, le programme s’arrête et renvoie la ligne ainsi que le type de l’erreur et le fichier “hexa.txt” n'est pas créé. Si une erreur mathématique est détectée, le programme vous renvoie l’instruction dans laquelle l’erreur est présente. Enfin, une fois toutes les erreurs corrigées, vous pouvez exécuter votre programme en assembleur. Bien sûr, notre programme ne vérifie pas la cohérence de celui entré.