Aide-Mémoire du débogueur GDB

1 Compilation d'un programme assembleur

- A l'aide d'un fichier makefile:
  `make`
  `make fichier.o`
  `make fichier.s`

2 Lancement et fin de GDB

- Pour lancer le débogueur, ouvrez un terminal puis : `make gdb`
- Au préalable lancez si nécessaire QEMU: `make qemu`
- Pour arrêter GDB et QEMU, sous GDB:
  `kill`
  `quit`
- Pour arrêter QEMU seul:
  `Ctrl-a puis c`
  `quit`

3 Exécution d'un programme sous GDB

- Pas à pas de l'assembleur
  `stepi` ou `si`: exécution d'une seule instruction.
  `nexti` ou `ni`: exécution d'une fonction complète.
- Pas à pas dans le code source
  `step` ou `s`: exécution d'une seule instruction.
  `next` ou `n`: exécution d'une fonction complète.
- Points d'arrêts
  `br nom_etiquette`
  `br nom_fichier.s:numero_ligne`
- Lancement
  `cont` : continue l'exécution jusqu'au prochain point d'arrêt.
  `run` : reprend l'exécution depuis le début.
- Arrêt complet
  `ctrl c`

4 Visualisation

- Du programme exécuté:
  `layout next` pour voir le prochaine forme de visualisation
  `layout regs` pour voir le contenu des registres
  `layout src` pour voir le code source
  `layout split` pour voir le code source et le code assembleur
  ...

- Des registres
  `print /base $nom_registre`

  où `base` peut être `x, d`, ... et `nom_registre` peut être `r1` ou `lr` ou `pc`
  On peut aussi écrire `p` au lieu de `print`.
  Par exemple `p /d $r4` donne la valeur du registre r4 en décimal.

- Du contenu de la mémoire
  `x /nombre format element adresse`
  où
  - `nombre` est le nombre d'élément à afficher
  - `format` est le format d'affichage : `x` pour hexadécimal, `d` pour décimal, `a` pour l'adresse, `c` pour le caractère.
  - `element` est la taille de l'entité: `b` pour l'octet, `h` pour le demi-mot, `w` pour le mot
  - `adresse` est l'adresse en mémoire à partir d'où on veut afficher les éléments. Cela peut être aussi le contenu d'un registre.
  - Exemples: `x /3xb 0x10004` affiche 3 octets en hexadécimal à partir de l'adresse `0x10004`
    `x /50xw $r13` affiche 50 mots de 4 octets en hexadécimal à partir de l'adresse contenu dans le registre `r13`.

- D'une étiquette (fonction...)
  `p nom_etiquette`

- D'une variable (du programme source)
  `p nom_variable`

- On peut aussi utiliser la commande `display` (avec les mêmes conventions que `print`) pour afficher un registre, une variable... à chaque fois que le programme s'arrête.

5 Modification

- Du contenu d'un registre ou d'une variable
  Il est possible de modifier pendant l'exécution d'un programme la valeurs d'un registres via GDB.
  `set nom=Valeur`
  où `nom` peut être un nom de registre (avec le `$`) ou un nom de variable.
  où `Valeur` peut être une étiquette, une valeur précise...
  Exemples : `set $r0=0x6A`
  `set $pc=_start`

6 Gestion de GDB

- Aide:
  `help nom_commande`