Séance machine assembleur Intel.

1 Accès à votre environnement

Le travail en machine s'effectue sur le serveur *ensilogic* auxquels vous pouvez vous connecter (*rlogin ensilogic*) depuis votre compte habituel sur *telesun*. Votre mot de passe intial sur *ensilogic* est : *psys2000*. La première chose à faire est de changer ce mot de passe initial par un mot de passe qui vous soit propre à l'aide de la commande : *passwd*.

Votre arborescence de fichiers sur *telesun* est montée sur *ensilogic*. Vos fichiers seront donc sauvegardés. En revanche, la charge du réseau risque d'être très importante et il vous est recommandé de ne pas vous connecter tous à la fois. Si vous souhaitez ouvrir plusieurs fenêtres (voir ci-dessous), il est recommandé d'ouvrir d'abord ces fenêtres sur *telesun* puis de vous connecter ensuite (*rlogin*) sur chacune d'entre elle à *ensilogic* ; la charge de travail sera ainsi mieux répartie entre les deux serveurs.

Pour des raisons évidentes de classement, il vous est recommandé de créer un répertoire particulier au logiciel de base pour y travailler durant les séances de TP. Vous pourrez imprimer des fichier au moyen des commandes Unix *lpr* ou *a2ps* en leur fournissant le nom d'une des imprimantes de l'étage.

Vous pouvez utiliser les éditeurs *vi*, *emacs* ou encore *nedit*.

2 Le travail à effectuer

La séance s'articule autour de programmes assembleur élémentaires testant les différents modes d'adressage sur Pentium et calculant le pgcd de deux nombres.

Vous allez d'abord recopier dans votre catalogue courant depuis le catalogue ˜chassin/ le fichier manip1.tar. Ceci peut être réalisé en tapant la commande Unix :

cp ˜chassin/manip1.tar ˜/.

Puis vous allez restaurer à l'aide du fichier recopié, un environnement de travail. Pour cela vous entrez la commande :

tar xvf manip1.tar

Cette commande crée dans le catalogue courant un catalogue *MANIP1* qui contient les fichiers : *Makefile, addon.s, adcode.s, pgcd.s* contenant les programmes assembleur permettant de tester les différents modes d'adressage de la mémoire et des données (voir polycopié au chapitre 3) et de calculer le pgcd des nombres 1024 et 256.

Makefile est le fichier qui, associé à la commande *make addon* (resp. *make adcode, make pgcd*) permet de génèrer dans le catalogue courant la liste d'assemblage *addon.l* (resp. ...), d'assembler *addon.s* et de construire un exécutable à partir du résultat de l'assemblage de *addon.s*. Le binaire exécutable produit a pour nom *addon*.

2.1 Test des modes d'adressage de la mémoire des données

Il s'agit de tester les modes d'adressage décrits dans la section 3.1 du polycopié de Xavier Rousset de Pina, pages 8 à 11.

1. Lisez les fichiers *addon.s* et *Makefile* qui sont a priori suffisamment commentés.
2. Produisez l'exécutable *addon* en entrant la commande *make addon*.
3. Ouvrir une fenêtre et y afficher *addon.l* au moyen des commandes Unix *less, more* ou *cat* ; ou affichez le au moyen de votre éditeur favori.

4. En utilisant les pages du polycopié décrivant les fonctions de base du metteur au point *gdb* exécutez le programme sous le contrôle de *gdb*. Pour cela on rappelle :
   (a) On tape la commande **gdb addon**
   (b) On place un point d'arrêt sur la première instruction du programme (étiquetée *main*) : **br main**
   (c) Affichez 20 lignes du programme à l'aide de la commande **x/20i main** et comparez rapidement le code désassemblé avec le code généré par l'assembleur dans le listing.
   (d) On lance le programme au moyen de la commande **run** : il va s'arrêter à la troisième instruction derrière le point d'entrée réel de *main*. Les deux premières instructions réalisent la liaison du programme avec son environnement et sont donc considérées comme indissociables de l'appel à *main* par *gdb*.
   (e) Vous exécutez le programme pas à pas (**si**) et regardez les valeurs des registres (**i reg**) et plus généralement vous entraînez aux commandes de *gdb*.
   (f) Les deux dernières instructions (*leave, ret*) redonnent le contrôle à l'environnement. Il n'y a plus lieu de continuer en "pas à pas" et il vaut mieux relancer l'exécution (**cont**).

2.2 Test des modes d'adressage de la mémoire des programmes

Il s'agit de tester les modes d'adressage décrits dans la section 3.2 du polycopié de Xavier Rousset de Pina, pages 11 à 13.

On prépare les fichiers *adcode.l* et *adcode* puis on appelle *gdb* de façon analogue au § 2.1. De la même façon on pose un point d'arrêt au début du programme. La suite diffère car, si là encore l'objectif est d'exécuter le programme pas à pas et d'examiner son comportement, cela ne sera possible qu'en modifiant constamment le pointeur d'instructions, à savoir le registe EIP. En effet, tous les tests faisant boucler le programme, il sera nécessaire de changer la valeur du registre EIP après chacun d'entr'eux, pour exécuter l'instruction suivante du programme : **set $eip=0x804839d**, si `0x804839d` est l'adresse de ladite instruction (repérée en exécutant la commande **x/15i main**). Après avoir vérifié (quelques instructions exécutées pas à pas) que l'instruction suivante entrainait le programme dans une boucle infinie, il faudra procéder à un nouveau chargement de la valeur de EIP pour exécuter l'instruction suivante.

Le branchement à l'adresse `0x080483CC` pose un problème particulier car celle-ci n'appartient pas à la mémoire du programme *adcode*. Après avoir vérifié la valeur de EIP au moyen de l'instruction *gdb* : **i reg eip**, il faut charger "à la main" l'adresse de l'instruction suivante et terminer l'exécution du programme comme au §2.1.

2.3 Calcul du pgcd de 256 et 1024

Pour produire les fichiers *pgcd.l* et *pgcd* puis lancer et arrêter le programme à son début sous *gdb*, on procédera de façon analogue aux §2.1 et §2.2. Contrairement au §2.2, il est possible d'exécuter l'ensemble du programme pas à pas. Cependant après exécution d'une itération de la boucle pas à pas, il est possible d'accélérer le déroulement du programme tout en suivant son évolution en insérant deux nouveaux points d'arrêt (**br iter** et **br fin**). Une fois ces points d'arrêts posés on relance l'exécution par la commande *gdb* **cont**. Il est ainsi possible de connaitre le résultat du calcul lors de l'exécution du second d'entre eux. Il est aussi possible de calculer d'autres valeurs de pgcd en modifiant les registres concernés à l'emplacement approprié ou encore de modifier le programme source *pgcd.s* afin d'effectuer, de façon analogue, d'autres calculs de pgcd.