ARCHITECTURE DES ORDINATEURS

TP : 05

Un peu de vraie vie

Pour voir à quoi peut ressembler la vraie vie, nous allons utiliser un émulateur de PC complet.

Vérifiez que cela fonctionne

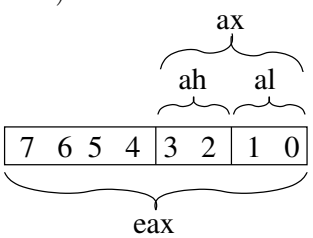
Copiez le répertoire `/net/cremi/sathibau/Archi/qemu` dans votre *home*. Lancez `make run` . Vous devriez obtenir une fenêtre avec juste un `A` et un `B` affichés en haut à gauche, et un C dans le terminal où vous avez lancé le `make run`. La fenêtre représente l'écran du PC virtuel, et le terminal représente la sortie du port série. Dans le terminal où vous avez lancé le `make run`, tapez une lettre (i.e. elle est envoyée sur le port série du PC virtuel). Celle-ci devrait s'afficher à l'écran. Attention, si par hasard vous cliquez sur la fenêtre qemu, vous ne pourrez plus utiliser la souris. Cf le titre de la fenêtre qemu : il faut appuyer sur `ctrl+alt` pour récupérer la souris.

Les pinailleurs diront qu'on est encore dans un émulateur, pas une vraie machine. Ils pourront essayer de graver le fichier iso et booter une vraie machine avec ;)

À la découverte de `monprog.S`

Ouvrez le fichier `monprog.S`, lisez et comprenez. Quelques clés de compréhension :

- Il s'agit d'assembleur x86, c'est-à-dire celui utilisé par les PC standards, l'y86 que nous avions appris jusqu'ici était une version simplifiée, ici c'est l'assembleur x86 complet.
- Les commentaires ici sont comme en C.
- Ici il n'y a plus de préfixes `mr`, `rr`, `ir` etc., et pour distinguer entre `i` et `m` on met un `$` devant les nombres et étiquettes pour le cas `i`. Ainsi, `movl 1,%eax` va lire en mémoire à l'adresse 1 ! Pour mettre 1 dans `%eax`, il faut utiliser `movl $1,%eax` . De même, `movl a,%eax` va mettre dans `%eax` le contenu de la mémoire à l'étiquette `a`, et `movl $a,%eax` va mettre dans `%eax` l'adresse de l'étiquette `a`.
- Ici en plus de `l` on utilise les suffixes `b` et `w` qui travaillent respectivement sur 8 bits et 16 bits. Il faut alors changer de nom de registre : par exemple, au lieu de `eax`, on pourra utiliser `al` et `ax`, qui sont respectivement les parties 8 bits et 16 bits du registre `eax`. Cela reste le même registre : écrire dans `eax` écrase le contenu de `al` (qui contient donc alors les 8 bits les plus faibles de la valeur 32 bits que l'on a écrite dans `eax`).



- L'écran est à l'adresse `0xb8000`, représenté par une matrice de 25x80x2 octets : 25 lignes de 80 colonnes, et un octet pour le caractère (ASCII, utilisez `man 7 ascii`) et un octet pour la couleur. Le premier caractère est donc à l'adresse `0xb8000`, le deuxième caractère est à l'adresse `0xb8002`, etc., le premier caractère de la deuxième ligne est à l'adresse `0xb80a0`, etc.

— Les instructions `inb` et `outb` permettent de lire et écrire sur les *ports d'entrée/sortie* qui pilotent les périphériques externes au processeur. Ici, on utilise seulement le port série : le port d'entrée/sortie `0x3F8` permet à la fois d'envoyer et de recevoir des caractères vers et depuis le port série. En réception, il faut cependant attendre que le bit de poids 0 du port `0x3F8+5` soit passé à 1.

Notes sur le débuggage

Pour le débuggage, on pourra utiliser `gdb` : lancez `make run-dbg`, cela ouvre une deuxième fenêtre avec gdb dedans. Lorsque l'exécution se suspend (ou lorsqu'on l'interrompt avec ctrl-C), vous pouvez utiliser `info registers` pour voir tous les registres ou juste `p $eax` pour examiner un registre, `s` (step) pour exécuter le programme pas à pas, `c` (continue) pour continuer l'exécution, etc.

À vous de jouer

Pour travailler, copiez l'exemple sous un autre nom :

```
$ cp monprog.S monautreprog.S
```

et pour lancer cet autre programme, utilisez `make run TORUN=monautreprog`

— Affichez les chiffres de 0 à 9 (attention, vérifiez bien le format de l'écran et évitez de mettre 0 comme couleur (c'est le code pour noir sur fond noir) !)
— Faites afficher à l'écran l'un derrière l'autre les caractères qui proviennent du port série [1]. Pour aller plus loin : gérez la touche entrée (caractère 13).
— Vous disposez de l'instruction `shr i,reg` qui *décale à droite* le registre `reg` de i bits, i.e. qui divise `reg` par 2^i. Écrivez une fonction prenant en paramètre un entier et qui l'affiche en hexadécimal à l'écran.

Pour aller plus loin...

— Utilisez cette fonction pour afficher le code ASCII des touches que vous tapez. Repérez celui de la touche `backspace` (effacement arrière), implémentez son fonctionnement.
— Faites déplacer un bonhomme (caractère 1) à l'écran avec les touches qsdz. Attention aux bords ! Faites-le se déplacer tout seul (vecteur vitesse 1,1), en le faisant rebondir sur les bords.

1. Oui, c'est normal que les lettres accentuées produisent un affichage étrange, on ne traitera pas ce problème.