COREWAR

Documentation

Par Platel_k, Guiho_r, Leprov_a et Decene_a.

Sommaire:

1. l'ASM

1.1 – Qu'est-ce que c'est?

L'ASM est le diminutif pour assembleur qui est un langage très bas niveau utilisé pour rendre lisible le langage machine pour l'homme. Il est représenté sous forme de mnémoniques qui sont de mot facile à retenir qui seront transformés en combinaison de bits différentes selon la mnémonique à représenter.

<IMAGE EXEMPLE .S>

1.2 – Structure d'un fichier .s.

Un fichier .s est ficher contenant ce fameux langage assembleur. Il se compose ici d'un '.name' qui définie le nom du fichier, d'un '.comment' qui permet d'indiquer un commentaire sur le programme. Il peut contenir un '.extend' si le programmeur veut pouvoir utiliser certaine autre possibilités du langage.

Ensuite vient le code à proprement parler. Il suit toujours le même schéma, c'est à dire:

```
Nom_du_label:    mnémonique parameter_1, parameter_2…
                 mnémonique parameter_1, parameter_2…
                 …
Nom_du_label:    mnémonique parameter_1, parameter_2…
                 mnémonique parameter_1, parameter_2…
                 …
```

1.3 – L'Objectif.

L'objectif de la première partie du projet est de réaliser une sorte de mini-compilateur pour transformer un .s, du langage assembleur en .cor, du langage machine.

<image .s> ➔ <image .cor>

1.4 – Le header type.

Le header type d'un .s se présente sous la forme :

```
.name « le nom du programme »
.comment « le commentaire du programme »        # pas obligatoire
.extend                                         # pas obligatoire
```

En .cor l'on retrouvera :

Le code magique

Le nom sur NAME_SIZE octets.

La taille totale des instructions en octets.

Le commentaire sur COMMENT_SIZE octets.

1.5 – Les types de valeur.

1.5.1 – Les directes.

C'est une valeur qui représente un nombre sur quatre octet.

Cette valeur est représentée par un modulo.

Exemple :

%42 -> 0000 002a

Symboliser par '10' dans l'octet de paramètre.

1.5.2 – les indirectes / index.

C'est une valeur qui représente souvent représente une adresse, c'est-à-dire le nombre d'octet que va devoir sauter le curseur pour arriver à l'adresse pointer.

Cette valeur est représentée par un nombre seul.

Exemple :

42 -> 002a

Symboliser par '11' dans l'octet de paramètre.

1.5.3 – les registres.

C'est une zone mémoire d'une taille définie par la machine virtuel. Lors de l'interprétation par la VM elle sera alors remplacer par la valeur contenue dans le registre au moment de la lecture.

Elle est symboliser par un 'r' suivie du numéro du registre.

Symbolisé par '01' dans l'octet de paramètre.

1.5.4 Cas particulier, les labels.

Les labels sont des variables qui représentent le nombre d'octet pour atteindre la première mnémonique du label.

Elle se représente sous la forme ':le_nom_du_label' sera l'adresse représenté sur 2 octet alors que '%:le_nom_du_label' sera l'adresse représenté sur 4 octet.

1.6 – L’octet de paramètre.

Il est présent après certaine instruction pour préciser à la VM quel type de paramètre elle va devoir traiter.

Il est codé sur 1 seul octet où chaque duo de bit représente le type de donné.

Ainsi par exemple un octet de paramètre valant en hexadécimal 90 donnera :

10 <- signifiant que le premier paramètre est un type direct.

01 <- signifiant que le deuxième paramètre est un type registre.

00 <- signifiant qu’il n’y a pas de paramètre

00 <- signifiant qu’il n’y a pas de paramètre

```
0x01 : live (alive)
-------------------
Description : Signal que le joueur est en vue.
            Prend une valeur sur DIR_SIZE octets.

Paramètres :
1 - Direct

Exemple :
         live %1
         0100000001
         Le joueur 1 est en vie.

Par défaut : 1 soi-même (supposition)
```

```
0x02 : ld
-----------
Description : Stock la valeur du premier paramètre dans un registre
passé en second paramètre.

Paramètres :
1 - Direct/Indirect
2 - Registre

Exemple :
         ld 34,r3
         02d0002203
         Charge la valeur contenue à l'adress 34 dans le registre r3.
```

0x03 : st (store)

Description : Charge la valeur du premier paramètre dans le second.
            Le premier paramètre est forcement un registre.

Paramètres :
1 - Registre
2 - Registre/Indirect/Direct

Exemple :
         st r1, :li1+1
         0370010020
         Charge la valeur contenue dans r1, dans le label à l'adresse
0020 (label li1 + 1).

_____
_____

0x04 : add (addition)

Description : Additionne les valeurs des deux premiers registre.
            Stock le résultat dans le troisième.
            Les trois paramètres sont des registres.

Paramètres :
1 - Registre
2 - Registre
3 - Registre

Exemple :
         add r3 r2 r5
         0454030205
         Additionne les registre r3 et r2 et place le résultat dans r5.
_____
_____

0x05 : sub (substraction)

Description : Même chose que add mais soustrait.

_____
_____

0x06 : and (et)

Description : Il applique l'opération binaire "&" au deux premiers
paramètres
            et stock le résultat dans le dernier paramètre.
            Le dernier paramètre est forcement un registre.

Paramètres :
1 - Direct/Indirect/Registre
2 - Direct/Indirect/Registre
3 - Registre

Exemple :
         and %20, 42, r3
         05b400000014002a03

Applique l'operateur binaire "&" à la valeure 20 et la valeure
contenue à l'adresse PC + 42 % IDX_MOD. Place le résultat dans
r3.

0x07 : or (ou)

Description : Même chose que and mais avec l'operateur binaire "|".

0x08 : xor (ou exclusif)

Description : Même chose que and mais avec l'operateur binaire "^".

0x09 : zjmp (zjump)

Description : Fait un saut à l'index passé en paramètre.
Opère seulement si carry vaut 1.
Si carry vaut 0, ne fait rien mais consomme le temps du
cycle.
Il n'y a pas d'octet indiquant la nature de paramètres
après.
Le paramètre est forcement un index.

Paramètres :
1 - index

Exemple :
zjump %:cp
09ffdb
Fait un saut à l'adresse PC + 0xffdb % IDX_MOD.

0x0a ldi (load index)

Description : Lit IND_SIZE octets à l'adresse PC + (premier_param %
IDX_MOD)
ajoute la valeur du second paramètre à cette valeur place
le résultat dans le registre indiqué en dernier paramètre.

Paramètres :
1 - Direct/Indirect/Registre
2 - Registre/Direct
3 - Registre

Exemple :
ldi %:lol, %10, r1
0aa40007000a01
Lit IND_SIZE octets à l'adresse PC + 7 % IDX_MOD ajoute 10 à
cette valeur
et les place dans le registre r1.

0x0b : sti (store index)

Description : Additionne le second et le troisième paramètre, et place le
contenu du registre à l'adresse indiquée par le résultat.
Le premier paramètre est forcement un registre.
Les deux derniers paramètres sont interprétés comme des index.

Paramètres :
1 - Registre
2 - Indirect/Direct/Registre
3 - Indirect/Registre

Exemple :
sti r2,42,%5
0b7802002a0005
Place la valeure de r2 à l'adresse 47.

0x0c : fork

Description : Crée un nouveau programme qui s'éxecute à partir de l'index passé en paramètre.
Prend les deux derniers octet de la valeur passé en paramètre.

Paramètres :
1 - Direct

Exemple :
fork %42
0cc000a2
Fork à l'adresse 42 % IDX_MODE + PC

0x0d : lld (long load)
0x0e : lldi (long load index)
0x0f : lfork (long fork)

Déplace de la même valeur sans le %IDX_MOD. L'opération modifie le carry.

0x10 : aff

Description : Equivalent d'un putchar.
Affiche sur la sortie standard le caractère ASCII contenue dans le registre passé en paramètre.
Fait un %256 pour s'assurer de rester dans la table ASCII.

Paramètres :
1- Registre

Exemple :
aff r3
104003
Affiche le caractère contenue dans r3 sur la sortie standard.