introduction

Jump to bottom

VBrazhnik a modifié cette page on Jan 3, 2019 · 1 révision

Une petite excursion dans l'histoire

En 1984, un article d'Andrew Dewdney est paru dans le magazine Scientific American décrivant le jeu **Core Wars** . L'article a commencé par les mots suivants:

> Deux programmes dans leur habitat naturel - la mémoire de l'ordinateur - se poursuivent d'adresse en adresse. Parfois, ils traquent l'ennemi; parfois des batteries de bombes numériques sont posées parfois, ils se copient dans un autre emplacement de mémoire pour éviter le danger, ou s'arrêtent pour réparer les dommages causés par l'ennemi. J'appelle ce jeu "Core Wars" ...

Source: Chapitre 2. Partie 1. Bataille sur "noyaux ferromagnétiques" ou MARS - le dieu de la guerre.

Présentation du projet "Corewar"

C'est sur la base de ce jeu assez populaire en cercles étroits que se construit le projet **Corewar** , qui se compose de trois parties obligatoires:

- Le champion
- Assembleur
- La machine virtuelle

Champion

L'essence de cette section est d'écrire du code en langage assembleur, qui sera ensuite placé dans un fichier avec l'extension `.s` .

> Nous écrivons actuellement du code dans un langage pseudo-assembleur. Autrement dit, dans un langage créé spécifiquement pour cette tâche, qui est similaire à un véritable assembleur, mais qui ne l'est toujours pas. Mais par souci de cohérence avec le texte de l'affectation, de simplicité et afin de sauver six lettres, nous appellerons également cet assemblage de langage.

Le code généré est notre champion, dont le but est de combattre d'autres champions, écrits par nous ou d'autres.

Le code de chaque champion a la structure suivante:

1. Nom
2. Commentaire
3. Code exécutable

Par exemple, cela pourrait ressembler à ceci:

```
.name       "Batman"
.comment    "This city needs me"

loop:
        sti r1, %:live, %1
live:
        live %0
        ld %0, r2
        zjmp %:loop
```

Dans ce projet, nous n'avons pas pour objectif de créer le champion le plus fort et le plus invincible. C'est un défi pour un projet complètement différent appelé le **"Corewar Championship"** .

Chez **Corewar,** nous créons notre champion uniquement pour démontrer la compréhension du sujet et la capacité à écrire du code en langage assembleur. Et pas pour qu'il puisse vaincre quelqu'un.

Notre tâche est d'écrire le code sans erreur afin que le programme `asm` puisse le transformer en bytecode, que la machine virtuelle exécuterait ensuite.

L'objectif de gagner la bataille ou de montrer au moins un résultat décent dans la bataille n'est pas devant nous.

Assembleur

Le but de cette section est de créer un programme qui traduira le code du champion écrit en langage assembleur en bytecode - un tas de nombres en système hexadécimal.

> Depuis elfique, cette tâche peut être interprétée comme "traduire des commandes d'un langage compréhensible à une personne (assembleur) dans un langage compréhensible par une machine virtuelle (byte-code)".

> **Traduction de programme** - conversion d'un programme présenté dans l'un des langages de programmation en un programme dans une autre langue. Le traducteur effectue généralement des diagnostics d'erreur, génère des dictionnaires d'identificateurs, imprime le texte du programme, etc.

Source: Traducteur - Wikipédia

Autrement dit, nous devons créer un programme nommé `asm` (du mot «assembleur»), qui recevra en paramètre un fichier avec du code en langage assembleur et créera un nouveau fichier avec du code d'octet basé sur celui-ci.

Le fichier avec le code de notre champion, écrit en assembleur, doit avoir l'extension `.s`. Sur cette base, le programme `asm` créera un nouveau fichier avec l'extension `.cor`, où se trouvera le bytecode généré.

Le nom du fichier lui-même restera inchangé. Autrement dit, après avoir appelé la commande `./asm batman.s`, un fichier `batman.s` doit apparaître à côté du fichier `batman.cor`. Bien sûr, si aucune erreur ne se produit lors de la diffusion.

Machine virtuelle

Après avoir reçu le fichier avec le bytecode, il est temps que la machine virtuelle s'exécute.

Une machine virtuelle est également un programme dont le fichier exécutable doit être nommé `corewar`.

Sa tâche est d'allouer une certaine section de mémoire, de placer sur cette section le code des champions et des voitures qui l'exécuteront.

Et puis suivez la bataille pour annoncer le champion gagnant une fois que c'est fini.

Bonus

Comme toujours, le nombre de bonus et leur essence sont limités uniquement par l'imagination de l'auteur.

En voici quelques-unes qui pourraient être implémentées:

Rapports d'erreur étendus

Si une erreur survient lors de la génération du bytecode, alors il est souhaitable que dans ce cas le programme `asm` se comporte comme un vrai traducteur, ce qu'il est vraiment. Autrement dit, il a affiché un message significatif - dans quelle ligne l' `.s` erreur s'est produite lors de l'utilisation du fichier et de quel type d'erreur il s'agissait.

Possibilité de démonter le bytecode

Rendez possible l'obtention du code source en langage assembleur en ayant un fichier bytecode à votre disposition. Autrement dit, implémentez la fonction opposée à celle pour laquelle le programme est destiné `asm`.

Visualiseur

Créez un programme qui affichera l'état de la mémoire et modifiera les paramètres clés du jeu pendant la bataille.

Différents effets sonores peuvent également s'y ajouter pour souligner des moments clés comme la mort de la calèche ou l'annonce du gagnant.

Liens utiles

L'Université des connaissances inutiles a une série en trois parties qui décrit les idées et les principes de Core Wars. Ces articles contiennent également des exemples de code de champion et une analyse de la façon dont ce code peut combattre et causer des dommages. Certes, tout cela est dans le langage du véritable assemblage.

Mais dans l'ensemble, c'est une bonne introduction au sujet, qui explique en détail ce qu'est l'essence de la «bataille en mémoire»:

- Chapitre 2. Partie 1. Bataille sur "noyaux ferromagnétiques" ou MARS - le dieu de la guerre.
- Chapitre 2. Partie 2. Bataille sur les "noyaux ferromagnétiques" ou le problème des jumeaux.
- Chapitre 2. Partie 3. Bataille sur les "noyaux ferromagnétiques" ou les processus de commutation.

▸ **Des pages** neuf

1. introduction
2. Fichiers fournis
3. Assembleur
4. Analyse lexicale
5. Assemblage en Bytecode
6. Démontage

7. Machine virtuelle
8. Visualisation

Cloner ce wiki localement

```
https://github.com/VBrazhnik/Corewar.wiki.git
```