Scale for project Corewar (/projects/corewar)

You should correct 4 students in this team

★

Git repository

vogsphere@vogsphere.42.fr:intra/2017/activities/corewar/lpousse

Guidelines

Ce barème contient des zones où VOUS devez décider, selon votre propre expérience, de quelle façon doit être contrôlée une fonctionnalité demandée.

Nous vous demandons, pour le bon déroulement de cette notation :

- De rester courtois, poli, respectueux, constructif, lors de cet échange. Le lien de confiance entre la communauté 42 et vous en dépend.
- De bien mettre en évidence auprès de la personne notée (ou du groupe) les dysfonctionnements éventuels.
- D'accepter qu'il puisse y a voir parfois des différences d'interprétation sur les demandes du sujet ou l'étendue des fonctionnalités. Restez ouvert d'esprit face a la vision de l'autre (a-t-il raison ? tort ?), et notez le plus honnêtement possible.

Vous avez normalement accès aux rendus effectués par la personne que vous devez noter. Récupérez-les pour effectuer la correction.

Bonne soutenance à tous !

Attachments

Subject (https://cdn.intra.42.fr/pdf/pdf/30/corewar.pdf) Resources (https://cdn.intra.42.fr/pdf/pdf/31/resources_corewar.pdf)

op.h (/uploads/document/document/27/op.h) op.c (/uploads/document/document/26/op.c)

Sections

Preliminaires

On vérifie les classiques.

Les trucs de base

Git clone, et correction UNIQUEMENT de ce qui se trouvait dans le repo.

On check :
- si il y a bien un rendu
- si le fichier auteur est conforme aux attentes du sujet
- si la norme est bien OK (verification avec la norminette)
- si il n'y a pas de cas de triche
- si le groupe est bien au complet

Si c'est pas OK, la notation s'arrete la.

Pour la suite, comme d'habitude, si un des 2 programme a un comportement qui n'est pas correct, on arrête la notation.

☑ Yes ☒ No

Consignes a lire

TL;DR

Prenez votre temps, don't be a dick, faites la soutenance en entier, et veillez à ce que correcteurs comme corrigés en ressortent grandis.

###############

Dans la vie, beaucoup de gens vivent selon des principes plus ou moins compliqués.

Dans cette soutenance en particulier, tout comme dans la vie, il est bon d'appliquer certains principes, notamment les suivants :

"Be rigorous"
"Don't be a dick"

Par là, il faut comprendre que :
* La rigueur, comme d'habitude, doit être une évidence;
* Cependant, ne soyez pas non plus MESQUIN, et concentrez vous sur l'évaluation honnête du travail du groupe plutôt que de chercher la petite bête.

Le but de cette soutenance n'est PAS de partir à la chasse aux zéros faciles.

Il y a beaucoup de choses à tester. Il va donc de soi que la soutenance va être longue. Ne prenez pas ça comme du temps de travail que vous perdez pour faire une correction, car faire les corrections fait partie de votre travail ... Si vous devez y passer deux heures, passez-y deux heures.

Comme on vous l'a dit pas mal de fois avant cette période de corrections, beaucoup de points du sujet sont soumis à interprétation, vous l'aurez constaté. Ce barème l'est donc également. Utilisez votre bon sens pour déterminer la justesse d'un point du rendu, et par dessus tout, appliquez le principe général "Don't be a dick".
Par exemple, une VM qui renomme les options demandées par le sujet, ou qui prend des champions avec une option devant pour préciser que c'est un champion, ou qui refuse de tourner avec 1 seul champion, ou ... EH BEN C'EST PAS GRAVE. Invalider une question pour quelque chose d'aussi TRIVIAL serait un exemple parfait de "being a dick". Pareil pour les affichages qui sont pas EXACTEMENT ceux du sujet, ou autres détails bêtes de ce genre. Utilisez votre bon sens pour déterminer ce qui constitue une faute et ce qui relève du simple détail cosmétique.

Toute personne qui fera cette soutenance à la va-vite sans se soucier de son impact sur les gens qu'elle corrige sera biflée en place publique. (c) zaz.

☑ Yes ☒ No

L'ASM

On va checker l'assembleur, hein.

Le .cor et son Header

L'assembleur doit générer au minimum un .cor avec un header valide.

Examiner la sortie avec hexdump -vC, pas diff, car certaines déviations dans la façon de gérer le nom ou le commentaire sont acceptables. Par exemple, générer un commentaire standard ou un nom par défaut pour un champion qui n'en a pas ferait dévier la sortie par rapport à l'assembleur de référence, mais ne constitue pas vraiment une ERREUR.

☑ Yes ☒ No

Le bytecode genere.

L'assembleur doit générer un bytecode valide pour le code du champion. Pas de déviation acceptable, s'il y en a une alors le champion généré n'est pas bon et ne peut pas tourner dans la VM. La, c'est delicat d'etre "cool", parcequ'un point qui n'est pas correct rend impossible l'execution du .cor dans une autre VM que la sienne (qui du coup gere mal le meme point).

- Les opcodes des instructions sont les bons. Ca c'est indispensable.

- Les octets de codage des paramètres, quand ils sont requis, sont présents et sont justes. Ils sont absents quand ils ne sont pas requis
- Les registres sont encodés sur 1 octet et leur valeur correspond à leur numéro
- Les indirects sont encodés sur 2 octets et leurs valeurs sont les bonnes (En big endian)
- Les directs sont encodés sur 4 octets pour les instructions ne prenant pas d'index, et leurs valeurs sont bonnes (En big endian)
- Les directs sont encodés sur 2 octets pour les instructions prenant des index, et leurs valeurs sont bonnes (En big endian)
- Les références vers labels sont converties vers les valeurs correspondantes (L'adresse d'un label est égale à l'adresse du premier octet de la ligne qui le suit, et une reference type %:labelX doit etre convertie vers (Adresse de debut de la ligne courante - (Adresse de debut de la ligne concernee par labelX))

Sur les 6 points precedents, une tolerence de 1 point qui n'est pas correct est acceptee pour valider cette question.

☑ Yes ☒ No

Gestion des erreurs

Les erreurs suivantes doivent être gérées dans l'ASM:
* Instructions inconnues
* Mauvais nombre de paramètres pour une instruction
* Mauvais types de paramètres pour une instruction
* Mauvais caractères dans un label
* Référence à un label inexistant depuis un direct ou un indirect
* D'une façon générale, toute erreur lexicale type mauvais caractere pour finir un label, un direct, ...

c'est OK ?

☑ Yes ☒ No

La VM

La base

La VM doit accepter de charger au moins un champion contenant un header valide, et reconnaitre les noms et les commentaires.
Ensuite, elle lance l'execution, et termine par afficher un message avec le joueur gagnant.

Yes No

L'execution des instructions

- La VM exécute correctement les instructions. Pour vérifier ça, vous pouvez utiliser quelques champions de test qui possèdent simplement une instruction (+ d'autres éventuellement pour "préparer le terrain"), et les exécuter avec l'option -dump sur la VM du groupe pour vérifier que la mémoire se retrouve dans le bon état.
On peut facilement vérifier l'état des registres en utilisant st, ou du carry en utilisant zjmp.

- La VM doit exécuter les processus dans le bon ordre, du dernier né jusqu'au premier né.
Typiquement, quand on lance une partie à deux champions, le process initial du deuxième champion s'exécute en premier.

Yes No

Les parametres speciaux de l'execution

- IDX_MOD
Les adresses d'action ont bien un modulo IDX_MOD qui leur est appliqué.

Concrètement cela concerne les choses suivantes :
* Toute lecture de la memoire pour determiner la valeur d'un indirect. (Pour l'indirect "516", on ne lit que a PC + 4 et pas a PC + 516)
* Premier parametre de "ld"
* Deuxieme parametre de "st"
* Parametre de zjmp
* Somme des deux premiers parametres de ldi
* Somme des deux derniers parametres de sti
* Parametre de fork

- Placement des champions
La VM doit répartir les champions correctement. Leurs points de depart doivent etre a egale distance les uns des autres (Modulo les arrondis si on a 3 joueurs).
Par exemple, pour deux champions sur une memoire de 4096 octets, les points de depart doivent etre espaces de 2048 octets. (Par exemple, le champion 1 peut démarrer à 0, et le champion 2 à 2048).

- Reglages du CYCLE_TO_DIE
La VM doit gérer les differents reglages de nombre de cycles correctement :
* Un processus qui ne fait pas au moins un live pendant une periode de CYCLE_TO_DIE doit mourir
* Si pendant une periode de CYCLE_TO_DIE il y a eu au moins NBR_LIVE live executes, au moment de la verification, la periode CYCLE_TO_DIE doit etre decrementee de CYCLE_DELTA
* Si lors d'une verification on constate qu'il n'y a pas eu de changement de CYCLE_TO_DIE depuis MAX_CHECKS verifications, alors on le decremente de toute facon.
* Si a l'issue d'une de ces verifications on constate que tous les processus sont morts, la partie est terminée.

Rappelez vous que le sujet ne précise pas si la vérification doit être faite en début ou en fin de cycle. Les deux interprétations sont valides.

Yes No

Gestion des options et des erreurs

La VM doit gérer les options demandées par le sujet : -dump et -n

Les erreurs suivantes doivent être gérées dans la VM
* Header invalide
* Pas de code
* Champion trop gros

Yes No

Et le reste

("surtout le reste" (c))

Organisation du groupe

Evaluez avec le groupe la façon dont ils se sont organises pour realiser ce corewar.
Plusieurs scenarios sont tout a fait recevables, soyez souples.
Ne validez pas cette question que si vous avez l'impression que le groupe a fait preuve de fouillis, de desorganisation chronique, d'une tres mauvaise gestion du temps.

Cette question est relativement subjective, et est bien sur comptabilisee comme telle. Mais le jugement subjectif d'une hierarchie fait partie de votre avenir professionnel, et meme si vous ne vous en rendez pas encore tous compte, vous pouvez agir dessus.

Yes No

Un champion

Le groupe doit avoir rendu un champion. Ce champion a simplement pour but de montrer que le groupe a compris comment écrire de l'ASM corewar, pas de battre quoi que ce soit.

Vérifiez donc simplement que le champion compile, et, si le coeur vous en dit, vérifiez qu'il bat au moins zork.s .

Yes No

Un gagnant

La VM trouve le bon gagnant, c'est à dire le champion qui a été ciblé par un "live" en tout dernier au moment où la partie se termine.

☑ Yes ☒ No

Encore quelques erreurs

Les erreurs suivantes doivent être gérées soit dans l'ASM soit dans la VM :
* Champion sans nom / sans commentaire
* Champion avec un nom ou un commentaire trop long
* Champion sans code

De nombreuses façons de gérer ces erreurs sont acceptables, comme par exemple :
* Refuser de compiler le champion
* Refuser de lancer le champion
* Corriger automatiquement le champion
* Lancer le champion même s'il n'a aucun code et le faire exécuter du vide
* ...

Ce qui compte ici, c'est que la façon des programmes de gérer les erreurs soit justifiée et ne soit pas complètement stupide. Faites preuve de bon sens.

☑ Yes ☒ No

Bonus

Plein de trucs

Plein de bonus sont possibles.

- super lexer / parser
- desassembleur de .cor
- de joli messages d'erreur a la compil, avec la ligne affichee et un curseur pour dire ou est le probleme (comme CLANG)
- une interface de visualisation (integree a la VM, ou via pipe ou shm, ou par fichier, ...)
- mode debugger sur la VM (step-by-step, inspection mémoire/état des process ... de tres nombreuses possibilités)

Rate it from 0 (failed) through 5 (excellent)

Ratings

Don't forget to check the flag corresponding to the defense

✔ Ok | Empty work | Incomplete work | No author file | Invalid compilation | Norme | Cheat | Crash

Conclusion

Leave a comment on this correction

*** (required) Comment**

Finish correction