Projet LO22
Un jeu de guerre électronique

1 ou 2 personnes

Table des matières

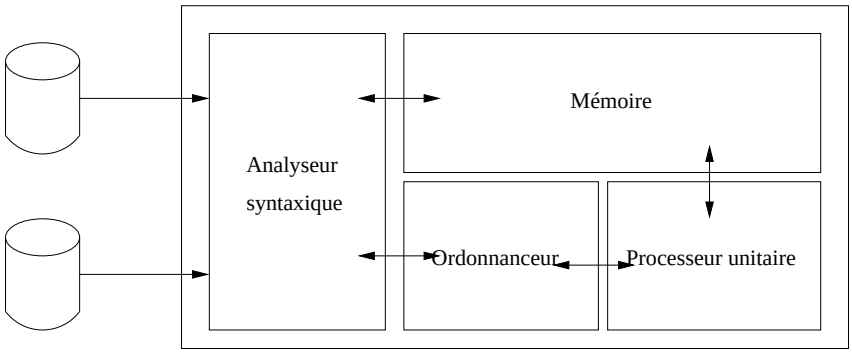
---

1 Sujet

But du simulateur

L'objectif de ce projet est de réaliser un simulateur de guerre électronique. Ce simulateur devra permettre l'opposition de deux programmes informatiques simples. Le perdant du jeu sera la programme qui réalise une instruction illégale, qui ne possède plus de mémoire... Chaque programme pourra élaborer des stratégies pour que ses concurrents réalisent le plus rapidement possible une erreur.

Chaque programme devra être écrit en un micro-assembleur (un language simple proche du language assembleur).



Language assembleur

Les programmes des compétiteurs seront écrits dans un language proche de l'assembleur. Les instructions sont les suivantes :

Code	Nom	Paramètres 1	Paramètres 2	Explication
0	`NOP`			Ne rien faire
1	`MOVE`	`adr`	`val`	Copie la valeur `val` dans la case mémoire numéro `adr`
2	`MOVA`	`adr1`	`adr2`	Copie la valeur se trouvant à l'adresse `adr2` dans la case mémoire numéro `adr1`
3	`ADD`	`adr`	`val`	Ajoute la valeur `val` à la valeur dans la case mémoire numéro `adr`
4	`ADDA`	`adr1`	`adr2`	Ajoute la valeur se trouvant à l'adresse `adr2` à la valeur dans la case mémoire numéro `adr1`

5	`CMP`	`adr1`	`adr2`	Compare les valeurs se trouvant dans les cases mémoire numéros `adr1` et `adr2`.
				`if (valeur(adr1)<valeur(adr2)) {`
				`valeur(adr1) = -1 ;`
				`}`
				`if (valeur(adr1)>valeur(adr2)) {`
				`valeur(adr1) = 1 ;`
				`}`
				`if (valeur(adr1)==valeur(adr2)) {`
				`valeur(adr1) = 0 ;`
				`}`

6	`JUMP`	`adr`		La prochaine instruction se trouve à ladresse mémoie `adr`
7	`JUMPIF`	`adr1`	`adr2`	Si la valeur se trouvant à ladresse mémoire `adr1` est différence de 0, alors la prochaine instruction se trouve à ladresse mémoie `adr`
8	`RANDOM`	`adr`		Tire un nombre au hasard entre 0 et 2^{32} et place cette valeur dans la case mémoire numéro `adr`
9	`END`			Arrête le programme

Le code d'une instruction est sa représentation sous forme de nombre entier. Il sera utilisé pour permettre le stockage en mémoire (voir ci-dessous).

Vous pouvez ajouter autant d'instructions que vous le désirez.

Mémoire

Le module de mémoire est une structure de donnée capable de stocker un ensemble de nombres entiers.

Elle a stocker des valeurs mais aussi des instructions. Les instructions sont toujours représentées par une suite de 3 entiers (3 cases mémoires) : la première contient le code de l'instruction, la seconde la valeur du premier paramètre et la troisième la valeur du second paramètre.

La mémoire doit permettre l'écriture et la lecture d'une case mémoire en fonction de sa position (son adresse) dans la mémoire. Les adresses de la mémoire commence à l'adresse 0.

Analyseur syntaxique

L'analyseur syntaxique est un module qui lit les deux fichiers contenant du code assembleur (par exemple : `MOVE 4 5`).

Ce module doit réaliser les tâches suivantes :

1. lire les deux fichiers
2. créer une représentation numérique du contenu de chaque fichier.
3. placer les deux représentations numériques dans la mémoire.
4. indiquer au module "ordonnanceur" les adresses des premières instructions à exécuter pour les programmes.

Processeur unitaire

Le processeur unitaire a pour rôle d'exécuter une seule instruction. L'ordonnanceur lui fournit l'adresse de l'instruction (c'est-à-dire l'adresse de l'entier contenant le code de l'instruction). Une fois cette information acquise, le processeur unitaire lit la mémoire et exécute l'instruction lu.

Le processeur unitaire doit indiquer s'il est arrivé ou non à exécuter l'instruction demandée. S'il a réussi à exécuté l'instruction demandée, il doit renvoyer le nouveau conmpteur ordinal. Ce nouveau compteur est égal à l'ancien compteur incrémenté de 3 (car une instructions est composée de trois cases mémoires) sauf pour les instructions `JUMP` et `JUMPIF` qui indiquent explicitement la valeur de ce compteur.

Ordonnanceur

L'ordonnanceur est le coeur du système. Pour chaque programme, il possède une information : le compteur ordinal. Un compteur ordinal est l'adresse de la prochaine instruction à exécuter. En d'autres termes, l'ordonnanceur connait la prochaine instruction de chaque programme.

Son rôle est de demander au processeur unitaire d'exécuter la prochaine instruction de chaque opposant tour à tour. Dès que le processeur unitaire ne peut pas exécuter une instruction, l'ordonnanceur arrête le jeu de guerre en proclammant l'autre programme vainceur.

2 Travail à faire

1. À partir des travaux réalisés en TD ou des codes sources fournis, implanter ou utiliser une bibliothèque C pour chaque module présenté ci-dessus (sauf l'analyseur syntaxique qui n'a pas été totalement présenté).
2. Implanter complètement l'analyseur syntaxique.
3. Écrire un fichier `Makefile` pour permettre la compilation de votre programme. Je ne compilerais pas "à la main" votre projet. S'il vous ne fournissez pas un `Makefile`, je considérerais que votre projet ne fonctionne pas.
4. Un rapport de maximum 5 pages dans l'un des formats suivants (par ordre de préférence) : **PDF**, **Postscript**, **HTML**, LaTeX, Word, Texte (pas de rapport manuscrit si possible).

3 Évaluation

J'évalurais votre projet selon les critères suivants (par ordre d'importance) :
- Maîtrise de l'algorithmie et de leur implantation système.
- Clarté et lisibilité de votre code source (choix des noms de fonctions et de variables, modularité...).
- Présence, utilité et clarté de vos commentaires.
- Qualité et efficience de votre `Makefile`.
- Esprit de synthèse dans votre rapport.

4 Directives

Directives devant être respectées :
- Tous ces fichiers devront être **obligatoirement** placées dans une archive (`.tar.gz`, `.zip`). Je n'accepterais aucun exécutable (uniquement des fichiers sources).
- Le rapport et l'archive de vos sources devront être envoyés à mon adresse email : `stephane.galland@utbm.fr`. Vous n'oublierez pas de mettre vos noms dans le sujet ou le corps de l'email (sinon il va partir à la poubelle sans que je l'ouvre).
- **DATE LIMITE POUR RENDRE VOTRE TRAVAIL** : 16/06/2006 à 18h00. Toute personne en retard verra son projet rejeté.