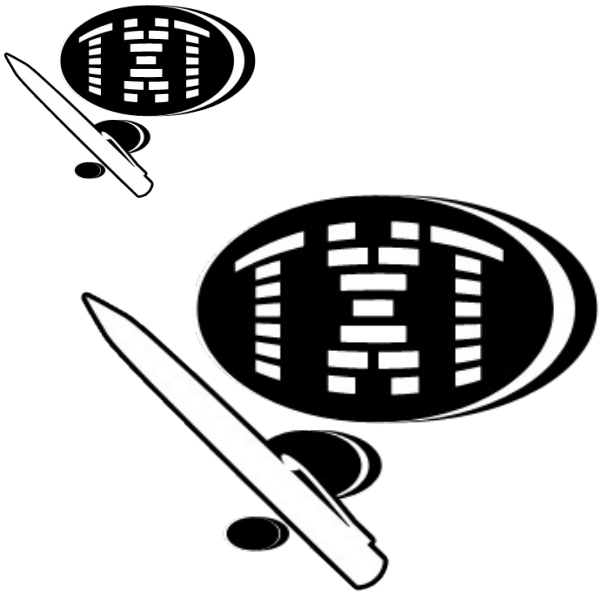

L A B T X T

D A E M O N L A B

MACRO RED ASSEMBLER

Documentation utilisateur pour le projet Editeur

- LabTxT -
- DaemonLab -

INDEX

01 – Détails administratifs

Votre travail doit être rendu via le dossier **~/projets/macrorasm/** dans votre espace personnel.

Pour cette activité, vous rendrez votre travail sous la forme d'un fichier au format PDF nommé **bpf.pdf**. Les sources ayant servit à générer ce PDF doivent être rendu avec votre travail.

Ce travail est a effectuer seul. Vous pouvez bien sur échanger avec vos camarades, néanmoins vous devez être l'auteur de votre travail. Utiliser le code d'un autre, c'est **tricher**. Et tricher annule **toutes** les médailles que vous avez reçu sur l'activité. La vérification de la triche est réalisée de la même manière que la correction : de manière **automatique**. Prenez garde si vous pensez pouvoir passer au travers.

02 – Macro RedAssembler

You will have to ***invent*** and then ***document*** a macro language for the RedSteel Assembly. Your document will be 2-3 pages long and features explaination about synctactic structures that should be useful to program the RedSteel microprocessor in an easier fashion.

Assembly is a pretty boring programming language in terms of representation : four columns, and that's it :

LABEL :
INSTRUCTION OPERAND1, OPERAND2

Meaning that structures like condition jump (if...) in high level langage *do not imply any difference in representation at all* !

ASSEMBLY	***C LANGUAGE***

`CMP VALUE1, VALUE2`
`IF< THEN, ELSE`
`THEN :`
`; POSITION A`
`SET ENDIF, PC`
`ELSE :`
`; POSITION B`
`ENDIF :`
	`IF (VALUE1 < VALUE2)`
	`{`
	`// POSITION A`
	`}`
	`ELSE`
	`{`
	`// POSITION B`
	`}`

In the same fashion, loops does not have the help of any kind of keyword. There is no while or for statement in assembly : it must be constructed with labels and jumps.

Also, some operations that are very self explanatory in programming langages are less explicit in assembly : A += 3 for example will be ADD #3, A in assembly. How would you improve that ?

Also, labels are pretty cool as they allow the programmer to define a custom name for a position in program, but there is no way to define a name for a memory space that is far from the program or for a register. Of course register usage changes a lot throught the program, but temporary name would be a nice thing to have, don't you think ?

Your documentation will talk about new **keywords** you will propose, but also if you have some, new **operators**. You may also of course add specific construction with specific symbol combination. Is there specific constructions you find appealing in C already ? Maybe some can be added...

You will also propose an implementation of your macro language, of tools you may or may not use to build it.