

D A E M O N L A B

REDASM

Une machine dans votre machine

- DaemonLab -

Ce projet consiste à réaliser un logiciel d'assemblage. Un logiciel d'assemblage est un logiciel assurant la transformation d'un langage de programmation, l'assembleur, en son équivalent binaire, exécutable par un ordinateur.

La machine pour lequel vous allez générer du binaire est un ordinateur 16 bits à l'architecture originale. La partie qui vous incombe est... son micro-processeur. Ce micro-processeur disposera de registres 16 bits, d'un bus de données de 16 bits, d'un bus d'adresse de 16bits et son byte ne sera pas un octet mais un seizet... soit 16bits et non 8.

Ce document est strictement personnel et ne doit en aucun cas être diffusé.



INDEX

Avant-propos :

- 01 - Détails administratifs
- 02 - Propreté de votre rendu
- 03 - Fonctions autorisées
- 04 - Micro-processeur ?
- 05 - Votre tâche
- 06 - L'assembleur
- 07 - L'émulateur
- 08 - L'architecture du REDSTEEL
- 09 - Étendre REDSTEEL
- 10 - Votre démo
- 11 - Projets annexes



01 – Détails administratifs

Votre travail doit être rendu le dossier ~/projets/redsteel/ dans l'espace personnel de l'administrateur de l'équipe.

Ce travail est à effectuer en équipe de 4. Durant ce projet, vous serez à même de travailler ensemble votre compréhension de la nature du micro-processeur autant que son implémentation. Lors de la soutenance finale, vous serez tous interrogé sur des aspects essentiels des ordinateurs afin de nous assurer de la compréhension de chacun

02 – Propreté de votre rendu

Votre rendu, c'est à dire le contenu de l'archive ou du dépôt que vous entrez sur l'interface du TechnoCentre, doit respecter strictement l'ensemble des règles suivantes :

- Il ne doit contenir aucun fichier objet. (*.o)
- Il ne doit contenir aucun fichier tampon. (*.~, ###)
- Il ne doit pas contenir votre production finale.

**La présence d'un fichier interdit mettra
immédiatement fin à votre évaluation.**

03 – Fonctions autorisées

Concernant le travail sur le CPU, vous n'avez le droit à aucune fonction.

Concernant un éventuel travail bonus, vous avez le droit aux fonctions suivantes :

- open, close, read, write, malloc, free, alloca
- srand, rand, cos, sin, atan2, sqrt
- opendir, readdir, closedir, mkdir, stat, getgrgid, getpwuid
- strftime, localtime, time



04 – Micro-processeur ? Émulateur ? Machine virtuelle ?

Qu'est ce qu'un processeur ? Il s'agit d'un système dont le métier consiste à réaliser des tâches, généralement simple, sur ordre. Qu'est ce qu'un micro-processeur ? Il s'agit d'un processeur fabriqué dans un micro-circuit, un « circuit intégré », dit « microchip » en anglais. C'est à dire une puce électronique.



Le Motorola 68000, un micro-processeur emblématique des années 80.

Les tâches à effectuer sont décrites dans un format appelé langage machine. Ce format est propre au micro-processeur. Vous pouvez voir le micro-processeur comme l'interprète de son propre langage machine. Une tâche porte le nom d'instruction.

Un émulateur est un logiciel réalisant la simulation d'un appareil. Votre terminal par exemple est en fait un émulateur de terminal car un terminal est un appareil électronique à l'origine :



Le VT100, un terminal produit par DEC, extrêmement répandu, permettant de se connecter à un « ordinateur central », c'est à dire un serveur, en mode texte.



Une « machine virtuelle » est donc l'effet produit par un émulateur lorsque celui-ci simule le fonctionnement d'un ordinateur.

Le terme « machine virtuelle » fait aussi souvent référence aux interprètes de langages binaires tel que la « Java virtual machine » ou JVM qui interprète le langage JAVA une fois celui-ci traduit en binaire.

Le terme est également utilisé par des logiciels effectuant une isolation d'une partie de l'environnement de l'ordinateur pour exécuter un environnement spécifique autre que celui d'exécution du logiciel isolant : VMWare, VirtualBox par exemple.

Dans le cadre du projet **REDASM**, vous allez devoir programmer un logiciel assurant la transformation du langage assembleur émulateur de micro-processeur. La description du langage est disponible parmi les ressources du projet.

Le fonctionnement du micro-processeur est de récupérer en mémoire une information : une instruction à exécuter, ainsi que les paramètres de cette instruction, normalement situé immédiatement après. Les instructions sont toujours simples : copier une valeur, faire une addition, déplacer la tête de lecture du programme, etc.

Le langage assembleur représente ces commandes en les nommant en lieu et place de les numéroter.

La documentation du **REDSTEEL** vous est fournie au format texte en annexe du projet, parmi les ressources, au même endroit que **rasm**, **remu** et que la carte vidéo.



05 – Votre tâche

Contrairement à d'autres projets où vous avez du tout réaliser, dans le projet **REDASM**, vous ne serez à l'origine que d'une partie du logiciel final.

Votre travail consistera à réaliser un logiciel d'assemblage fournissant des fichiers exploitables par **remu**, l'émulateur.

Le travail de **remu** étant de lier les différentes parties de la machine virtuelle, son micro-processeur, son clavier, sa carte vidéo, sa carte son, sa carte de gestion des disques et sa carte réseau. L'émulateur apporte la mémoire centrale ainsi que le mécanisme de chargement de deux fichiers **.red** contenant du code machine, qui eux sont produit par votre logiciel.

Les fichiers sont **rom.red**, contenant la ROM de la machine, et un autre fichier **.red** quelconque qui sera chargé comme « cartouche » à la manière d'une console de jeu.

L'émulation est complète mais il n'est pas possible d'exploiter ses mécanismes faute de programme : il manque l'élément permettant d'écrire, le logiciel d'assemblage, dont le rôle est de lire le code assembleur et de générer les fichiers binaires équivalents. Programmer cette partie là est votre rôle.

Pour vous aider, vous avez bien sur **remu**, mais également **rasm**, qui vous servira de programme de référence : en effet, **rasm** n'est rien d'autre que... le programme que vous devez écrire.

Ce n'est pas tout, vous avez à disposition également plusieurs fichiers **.rs** que vous pourrez exécuter sur votre micro-processeur afin d'en tester le fonctionnement. Vous disposerez également de la **TurboFX GeBansheeForce 2000RTX Voodoo**, une carte vidéo virtuelle compatible avec **remu** et qui vous permettra d'aller encore plus loin dans vos tests.

Pour terminer, vous disposerez également d'un micro-processeur virtuel conçu par le **DAEMONLAB** comme référence de fonctionnement. Vous êtes bien sur libre d'étendre votre propre micro-processeur tant qu'il demeure compatible (par exemple, en ajoutant des instructions, des registres ou toute autre chose que vous pourrez imaginer)



06 – L'assembleur

Le logiciel **rasm** pour **RedAssembler** est un logiciel permettant d'effectuer une traduction assembleur vers langage machine (opération appelé assemblage) et également l'opération inverse (appelé désassemblage)

Le logiciel **rasm** vous servira à générer des programmes pour **remu** et à vérifier le fonctionnement de votre propre implémentation de l'assembleur.

Notez bien que le désassemblage ne permet pas de résoudre tous les symboles du programme ! En effet, si vous avez crée des emplacements pour des données rangées derrière un nom, ce nom n'est jamais présent dans le programme final : l'endroit auquel il fait référence est directement utilisé. De même, les données seront désassemblé comme des instructions et non comme des données ! Bien entendu, les commentaires seront également perdus.

Par exemple, le programme suivant :

```
! BOUCLE INFINIE SUR UNE LIGNE  
START:  
SET #START, PC
```

Une fois assemblé, puis désassemblé, celui-ci donne le code suivant :

```
SET #0, [0]
```

Le symbole **start** étant présent au début du programme (à l'adresse 0 de celui-ci), sa valeur est donc de 0. Lors de l'assemblage, les appels fait à lui sont donc remplacé par sa valeur. L'opération inverse n'est pas possible car rien ne permet de savoir que la valeur était un label dans le code original.

L'opération **SET** ici met donc la valeur 0 dans le registre « **PC** » (Program counter). Un registre est un emplacement mémoire situé dans le micro-processeur. Le **Program Counter** contient l'adresse de la prochaine instruction qui sera exécuté. Ici, c'est très clair : la prochaine instruction, c'est celle-là, donc, comme le dis le commentaire, c'est une boucle infinie.

L'écriture d'une valeur n'étant pas un multiple de 3 dans le **Program Counter** provoque sa ré-écriture immédiate par le micro-processeur au multiple de trois qui suis immédiatement.



Outre les instructions du micro-processeur, le programme **rasm** vous permet de manipuler le binaire de sortie lorsqu'il assemble :

La directive **data** vous permet de placer des valeurs arbitraires à l'emplacement actuel dans le programme.

```
DATA    0, 1, 2 ; 3 ENTIERS
DATA    3,0, 3,1, 3,2 ; 3 FLOAT
DATA    "ABC" ; 3 CARACTÈRES
; 1 BYTE A 0B0001101100011011
DATA    '0123 =0X'
DATA    [32] ; 32 BYTES A 0
```

Les trois entiers sont chacun sur 1 byte.

Les flottants prennent chacun 2 bytes (partie entière, partie décimale) : la partie entière exploite la capacité du byte tandis que la partie décimale varie uniquement jusqu'à 10000.

La chaîne de caractère n'incorpore pas de terminateur nul et chaque caractère occupe un octet et non un byte.

Le symbole apostrophe sert à indiquer une plage de couples de bits. Les symboles 0123 valent respectivement... 0 1 2 et 3. Les symboles espace = 0 et X valent également 0 1 2 et 3. Chaque valeur occupe 2 bits seulement. Ce format est disponible afin de faciliter l'utilisation de la carte vidéo fournie par le laboratoire de programmation générale qui exploite des couleurs sur 2 bits.

Le symbole crochet gauche, suivi d'un entier et d'un crochet droite permet d'indiquer une plage de byte mis à zéro.

La transition depuis une plage de données vers une instruction provoque un réalignement de l'adresse d'écriture sur un multiple de 3.

La directive **add_offset** permet de sauter une quantité données de bytes passée comme unique paramètre. Les bytes sautés sont initialisés à zéro.

La directive **set_position** permet de sauter une quantité de données jusqu'à la position passée en paramètre, qui ne peut être inférieure à la position actuelle.

La directive **set_label_start** permet d'ajouter une valeur à tous les labels situés après la directive. Par défaut, cette valeur est 0. Un paramètre est requis. Cette directive est indispensable à la programmation de la cartouche.



07 – RedEmulator

Le logiciel **remu** pour RedEmulator est un logiciel permettant d'effectuer la jonction entre différentes bibliothèques logicielles dynamiques. Il apporte un panneau de débogage permettant d'afficher des valeurs de la mémoire de l'ordinateur, qu'il s'agisse d'instruction ou de données ainsi que bien sûr l'état du micro-processeur lui-même.

Les bibliothèques chargées par remu sont les émulateurs de CPU, les émulateurs des périphériques de saisie (clavier, souris, manette...), de carte vidéo, de carte son, de gestionnaire de disques et de carte réseau. Le **DAEMONLAB** vous fournit pour rappel un émulateur de carte vidéo. Cet émulateur permet un affichage à résolution variable en multi-couche. Plus la résolution est élevée, moins de couches sont disponibles. Sa documentation est avec les ressources du projet. Vous pouvez bien entendu réaliser vos propres périphériques. Une documentation est disponible parmi les ressources du projet.

Le programme **remu** exploite la ligne de commande et un fichier de configuration afin de régler les liaisons qui existent entre lui et les bibliothèques extérieures. Un exemple vous est fourni.

Concernant le débogueur, voici comment il fonctionne :

CURRENT REGISTERS										STATUS REG				STACK			
R0: 00000000	R1: 00000000	R2: 00000000	R3: 00000000	R4: 00000000	R5: 00000000	R6: 00000000	R7: 00000000	R8: 00000000	R9: 00000000	CR: 00000000	CF: 00000000	X0: 00000000	X1: 00000000	X2: 00000000	X3: 00000000		
R10: 00000000	R11: 00000000	R12: 00000000	R13: 00000000	R14: 00000000	R15: 00000000	R16: 00000000	R17: 00000000	R18: 00000000	R19: 00000000	FG: 00000000	DF: 00000000	X4: 00000000	X5: 00000000	X6: 00000000	X7: 00000000		
R20: 00000000	R21: 00000000	R22: 00000000	R23: 00000000	R24: 00000000	R25: 00000000	R26: 00000000	R27: 00000000	R28: 00000000	R29: 00000000	LB: 00000000	DB: 00000000	X8: 00000000	X9: 00000000	X10: 00000000	X11: 00000000		
R30: 00000000	R31: 00000000	R32: 00000000	R33: 00000000	R34: 00000000	R35: 00000000	R36: 00000000	R37: 00000000	R38: 00000000	R39: 00000000	CB: 00000000	DB: 00000000	X12: 00000000	X13: 00000000	X14: 00000000	X15: 00000000		

PROGRAM										MEMORY																																																																																																																																																																																																																																																																																																																																																																																																				
00000000: 80FE #1F08, \$D00	00000001: 80FE #10A0, --, \$DFF	00000002: RND #179C, [C4B1]	00000003: C4B1 [C01C]--, #473	00000004: DD #508, #FC6	00000005: DD #508, #FC6	00000006: 95D [8F06]--, #39D5	00000007: IF- [C091]--, #412	00000008: 92A2 #C0A9, #7130	00000009: 0FE9 [DC0B], \$D5B4	0000000A: 4DFE --[C6705]--, #F4BC	0000000B: H4FE [11AC]--, [C65C]	0000000C: D0ED [C02F]--, ++[C059]++	0000000D: XFE [D019], ++[C4175]	0000000E: CL #745B,	0000000F: 9374 #9FB, #9821	00000010: 9374 #9FB, #9821	00000011: 4413 #B205, #018	00000012: 9320 #B5E9, [CBAC2]++	00000013: 9350 #C72, #978D	00000014: IF0 #1300, ++[CF95]	00000015: XOR ++[C05C], ++[1042]	00000016: 9302 #9590, \$1D71	00000017: D076 #5950, ++[ED8B]++	00000018: 1B0E ++[CFD5C], [C61E5]	00000019: 1071 \$00E0, [0141]	0000001A: 80FE #1F08, \$D00	0000001B: 80FE #10A0, --, \$DFF	0000001C: RND #179C, [C4B1]	0000001D: C4B1 [C01C]--, #473	0000001E: DD #508, #FC6	0000001F: DD #508, #FC6	00000020: 95D [8F06]--, #39D5	00000021: IF- [C091]--, #412	00000022: 92A2 #C0A9, #7130	00000023: 0FE9 [DC0B], \$D5B4	00000024: 4DFE --[C6705]--, #F4BC	00000025: H4FE [11AC]--, [C65C]	00000026: D0ED [C02F]--, ++[C059]++	00000027: XFE [D019], ++[C4175]	00000028: CL #745B,	00000029: 9374 #9FB, #9821	0000002A: 9374 #9FB, #9821	0000002B: 4413 #B205, #018	0000002C: 9320 #B5E9, [CBAC2]++	0000002D: 9350 #C72, #978D	0000002E: IF0 #1300, ++[CF95]	0000002F: XOR ++[C05C], ++[1042]	00000030: 9302 #9590, \$1D71	00000031: D076 #5950, ++[ED8B]++	00000032: 1B0E ++[CFD5C], [C61E5]	00000033: 1071 \$00E0, [0141]	00000034: 80FE #1F08, \$D00	00000035: 80FE #10A0, --, \$DFF	00000036: RND #179C, [C4B1]	00000037: C4B1 [C01C]--, #473	00000038: DD #508, #FC6	00000039: DD #508, #FC6	0000003A: 95D [8F06]--, #39D5	0000003B: IF- [C091]--, #412	0000003C: 92A2 #C0A9, #7130	0000003D: 0FE9 [DC0B], \$D5B4	0000003E: 4DFE --[C6705]--, #F4BC	0000003F: H4FE [11AC]--, [C65C]	00000040: D0ED [C02F]--, ++[C059]++	00000041: XFE [D019], ++[C4175]	00000042: CL #745B,	00000043: 9374 #9FB, #9821	00000044: 9374 #9FB, #9821	00000045: 4413 #B205, #018	00000046: 9320 #B5E9, [CBAC2]++	00000047: 9350 #C72, #978D	00000048: IF0 #1300, ++[CF95]	00000049: XOR ++[C05C], ++[1042]	0000004A: 9302 #9590, \$1D71	0000004B: D076 #5950, ++[ED8B]++	0000004C: 1B0E ++[CFD5C], [C61E5]	0000004D: 1071 \$00E0, [0141]	0000004E: 80FE #1F08, \$D00	0000004F: 80FE #10A0, --, \$DFF	00000050: RND #179C, [C4B1]	00000051: C4B1 [C01C]--, #473	00000052: DD #508, #FC6	00000053: DD #508, #FC6	00000054: 95D [8F06]--, #39D5	00000055: IF- [C091]--, #412	00000056: 92A2 #C0A9, #7130	00000057: 0FE9 [DC0B], \$D5B4	00000058: 4DFE --[C6705]--, #F4BC	00000059: H4FE [11AC]--, [C65C]	0000005A: D0ED [C02F]--, ++[C059]++	0000005B: XFE [D019], ++[C4175]	0000005C: CL #745B,	0000005D: 9374 #9FB, #9821	0000005E: 9374 #9FB, #9821	0000005F: 4413 #B205, #018	00000060: 9320 #B5E9, [CBAC2]++	00000061: 9350 #C72, #978D	00000062: IF0 #1300, ++[CF95]	00000063: XOR ++[C05C], ++[1042]	00000064: 9302 #9590, \$1D71	00000065: D076 #5950, ++[ED8B]++	00000066: 1B0E ++[CFD5C], [C61E5]	00000067: 1071 \$00E0, [0141]	00000068: 80FE #1F08, \$D00	00000069: 80FE #10A0, --, \$DFF	0000006A: RND #179C, [C4B1]	0000006B: C4B1 [C01C]--, #473	0000006C: DD #508, #FC6	0000006D: DD #508, #FC6	0000006E: 95D [8F06]--, #39D5	0000006F: IF- [C091]--, #412	00000070: 92A2 #C0A9, #7130	00000071: 0FE9 [DC0B], \$D5B4	00000072: 4DFE --[C6705]--, #F4BC	00000073: H4FE [11AC]--, [C65C]	00000074: D0ED [C02F]--, ++[C059]++	00000075: XFE [D019], ++[C4175]	00000076: CL #745B,	00000077: 9374 #9FB, #9821	00000078: 9374 #9FB, #9821	00000079: 4413 #B205, #018	0000007A: 9320 #B5E9, [CBAC2]++	0000007B: 9350 #C72, #978D	0000007C: IF0 #1300, ++[CF95]	0000007D: XOR ++[C05C], ++[1042]	0000007E: 9302 #9590, \$1D71	0000007F: D076 #5950, ++[ED8B]++	00000080: 1B0E ++[CFD5C], [C61E5]	00000081: 1071 \$00E0, [0141]	00000082: 80FE #1F08, \$D00	00000083: 80FE #10A0, --, \$DFF	00000084: RND #179C, [C4B1]	00000085: C4B1 [C01C]--, #473	00000086: DD #508, #FC6	00000087: DD #508, #FC6	00000088: 95D [8F06]--, #39D5	00000089: IF- [C091]--, #412	0000008A: 92A2 #C0A9, #7130	0000008B: 0FE9 [DC0B], \$D5B4	0000008C: 4DFE --[C6705]--, #F4BC	0000008D: H4FE [11AC]--, [C65C]	0000008E: D0ED [C02F]--, ++[C059]++	0000008F: XFE [D019], ++[C4175]	00000090: CL #745B,	00000091: 9374 #9FB, #9821	00000092: 9374 #9FB, #9821	00000093: 4413 #B205, #018	00000094: 9320 #B5E9, [CBAC2]++	00000095: 9350 #C72, #978D	00000096: IF0 #1300, ++[CF95]	00000097: XOR ++[C05C], ++[1042]	00000098: 9302 #9590, \$1D71	00000099: D076 #5950, ++[ED8B]++	0000009A: 1B0E ++[CFD5C], [C61E5]	0000009B: 1071 \$00E0, [0141]	0000009C: 80FE #1F08, \$D00	0000009D: 80FE #10A0, --, \$DFF	0000009E: RND #179C, [C4B1]	0000009F: C4B1 [C01C]--, #473	000000A0: DD #508, #FC6	000000A1: DD #508, #FC6	000000A2: 95D [8F06]--, #39D5	000000A3: IF- [C091]--, #412	000000A4: 92A2 #C0A9, #7130	000000A5: 0FE9 [DC0B], \$D5B4	000000A6: 4DFE --[C6705]--, #F4BC	000000A7: H4FE [11AC]--, [C65C]	000000A8: D0ED [C02F]--, ++[C059]++	000000A9: XFE [D019], ++[C4175]	000000AA: CL #745B,	000000AB: 9374 #9FB, #9821	000000AC: 9374 #9FB, #9821	000000AD: 4413 #B205, #018	000000AE: 9320 #B5E9, [CBAC2]++	000000AF: 9350 #C72, #978D	000000B0: IF0 #1300, ++[CF95]	000000B1: XOR ++[C05C], ++[1042]	000000B2: 9302 #9590, \$1D71	000000B3: D076 #5950, ++[ED8B]++	000000B4: 1B0E ++[CFD5C], [C61E5]	000000B5: 1071 \$00E0, [0141]	000000B6: 80FE #1F08, \$D00	000000B7: 80FE #10A0, --, \$DFF	000000B8: RND #179C, [C4B1]	000000B9: C4B1 [C01C]--, #473	000000BA: DD #508, #FC6	000000BB: DD #508, #FC6	000000BC: 95D [8F06]--, #39D5	000000BD: IF- [C091]--, #412	000000BE: 92A2 #C0A9, #7130	000000BF: 0FE9 [DC0B], \$D5B4	000000C0: 4DFE --[C6705]--, #F4BC	000000C1: H4FE [11AC]--, [C65C]	000000C2: D0ED [C02F]--, ++[C059]++	000000C3: XFE [D019], ++[C4175]	000000C4: CL #745B,	000000C5: 9374 #9FB, #9821	000000C6: 9374 #9FB, #9821	000000C7: 4413 #B205, #018	000000C8: 9320 #B5E9, [CBAC2]++	000000C9: 9350 #C72, #978D	000000CA: IF0 #1300, ++[CF95]	000000CB: XOR ++[C05C], ++[1042]	000000CC: 9302 #9590, \$1D71	000000CD: D076 #5950, ++[ED8B]++	000000CE: 1B0E ++[CFD5C], [C61E5]	000000CF: 1071 \$00E0, [0141]	000000D0: 80FE #1F08, \$D00	000000D1: 80FE #10A0, --, \$DFF	000000D2: RND #179C, [C4B1]	000000D3: C4B1 [C01C]--, #473	000000D4: DD #508, #FC6	000000D5: DD #508, #FC6	000000D6: 95D [8F06]--, #39D5	000000D7: IF- [C091]--, #412	000000D8: 92A2 #C0A9, #7130	000000D9: 0FE9 [DC0B], \$D5B4	000000DA: 4DFE --[C6705]--, #F4BC	000000DB: H4FE [11AC]--, [C65C]	000000DC: D0ED [C02F]--, ++[C059]++	000000DD: XFE [D019], ++[C4175]	000000DE: CL #745B,	000000DF: 9374 #9FB, #9821	000000E0: 9374 #9FB, #9821	000000E1: 4413 #B205, #018	000000E2: 9320 #B5E9, [CBAC2]++	000000E3: 9350 #C72, #978D	000000E4: IF0 #1300, ++[CF95]	000000E5: XOR ++[C05C], ++[1042]	000000E6: 9302 #9590, \$1D71	000000E7: D076 #5950, ++[ED8B]++	000000E8: 1B0E ++[CFD5C], [C61E5]	000000E9: 1071 \$00E0, [0141]	000000EA: 80FE #1F08, \$D00	000000EB: 80FE #10A0, --, \$DFF	000000EC: RND #179C, [C4B1]	000000ED: C4B1 [C01C]--, #473	000000EE: DD #508, #FC6	000000EF: DD #508, #FC6	000000F0: 95D [8F06]--, #39D5	000000F1: IF- [C091]--, #412	000000F2: 92A2 #C0A9, #7130	000000F3: 0FE9 [DC0B], \$D5B4	000000F4: 4DFE --[C6705]--, #F4BC	000000F5: H4FE [11AC]--, [C65C]	000000F6: D0ED [C02F]--, ++[C059]++	000000F7: XFE [D019], ++[C4175]	000000F8: CL #745B,	000000F9: 9374 #9FB, #9821	000000FA: 9374 #9FB, #9821	000000FB: 4413 #B205, #018	000000FC: 9320 #B5E9, [CBAC2]++	000000FD: 9350 #C72, #978D	000000FE: IF0 #1300, ++[CF95]	000000FF: XOR ++[C05C], ++[1042]	00000100: 9302 #9590, \$1D71	00000101: D076 #5950, ++[ED8B]++	00000102: 1B0E ++[CFD5C], [C61E5]	00000103: 1071 \$00E0, [0141]	00000104: 80FE #1F08, \$D00	00000105: 80FE #10A0, --, \$DFF	00000106: RND #179C, [C4B1]	00000107: C4B1 [C01C]--, #473	00000108: DD #508, #FC6	00000109: DD #508, #FC6	0000010A: 95D [8F06]--, #39D5	0000010B: IF- [C091]--, #412	0000010C: 92A2 #C0A9, #7130	0000010D: 0FE9 [DC0B], \$D5B4	0000010E: 4DFE --[C6705]--, #F4BC	0000010F: H4FE [11AC]--, [C65C]	00000110: D0ED [C02F]--, ++[C059]++	00000111: XFE [D019], ++[C4175]	00000112: CL #745B,	00000113: 9374 #9FB, #9821	00000114: 9374 #9FB, #9821	00000115: 4413 #B205, #018	00000116: 9320 #B5E9, [CBAC2]++	00000117: 9350 #C72, #978D	00000118: IF0 #1300, ++[CF95]	00000119: XOR ++[C05C], ++[1042]	0000011A: 9302 #9590, \$1D71	0000011B: D076 #5950, ++[ED8B]++	0000011C: 1B0E ++[CFD5C], [C61E5]	0000011D: 1071 \$00E0, [0141]	0000011E: 80FE #1F08, \$D00	0000011F: 80FE #10A0, --, \$DFF	00000120: RND #179C, [C4B1]	00000121: C4B1 [C01C]--, #473	00000122: DD #508, #FC6	00000123: DD #508, #FC6	00000124: 95D [8F06]--, #39D5	00000125: IF- [C091]--, #412	00000126: 92A2 #C0A9, #7130	00000127: 0FE9 [DC0B], \$D5B4	00000128: 4DFE --[C6705]--, #F4BC	00000129: H4FE [11AC]--, [C65C]	0000012A: D0ED [C02F]--, ++[C059]++	0000012B: XFE [D019], ++[C4175]	0000012C: CL #745B,	0000012D: 9374 #9FB, #9821	0000012E: 9374 #9FB, #9821	0000012F: 4413 #B205, #018	00000130: 9320 #B5E9, [CBAC2]++	00000131: 9350 #C72, #978D	00000132: IF0 #1300, ++[CF95]	00000133: XOR ++[C05C], ++[1042]	00000134: 9302 #9590, \$1D71	00000135: D076 #5950, ++[ED8B]++	00000136: 1B0E ++[CFD5C], [C61E5]	00000137: 1071 \$00E0, [0141]	00000138: 80FE #1F08, \$D00	00000139: 80FE #10A0, --, \$DFF	0000013A: RND #179C, [C4B1]	0000013B: C4B1 [C01C]--, #473	0000013C: DD #508, #FC6	0000013D: DD #508, #FC6	0000013E: 95D [8F06]--, #39D5	0000013F: IF- [C091]--, #412	00000140: 92A2 #C0A9, #7130	00000141: 0FE9 [DC0B], \$D5B4	00000142: 4DFE --[C6705]--, #F4BC	00000143: H4FE [11AC]--, [C65C]	00000144: D0ED [C02F]--, ++[C059]++	00000145: XFE [D019], ++[C4175]	00000146: CL #745B,	00000147: 9374 #9FB, #9821	00000148: 9374 #9FB, #9821	00000149: 4413 #B205, #018	0000014A: 9320 #B5E9, [CBAC2]++	0000014B: 9350 #C72, #978D	0000014C: IF0 #1300, ++[CF95]	0000014D: XOR ++[C05C], ++[1042]	0000014E: 9302 #9590, \$1D71	0000014F: D076 #5950, ++[ED8B]++	00000150: 1B0E ++[CFD5C], [C61E5]	00000151: 1071 \$00E0, [0141]	00000152: 80FE #1F08, \$D00	00000153: 80FE #10A0, --, \$DFF	00000154: RND #179C, [C4B1]	00000155: C4B1 [C01C]--, #473	00000156: DD #508, #FC6	00000157: DD #508, #FC6	00000158: 95D [8F06]--, #39D5	00000159: IF- [C091]--, #412	0000015A: 92A2 #C0A9, #7130	0000015B: 0FE9 [DC0B], \$D5B4	0000015C: 4DFE --[C6705]--, #F4BC	0000015D: H4FE [11AC]--, [C65C]	0000015E: D0ED [C02F]--, ++[C059]++	0000015F: XFE [D019], ++[C4175]	00000160: CL #745B,	00000161: 9374 #9FB, #9821	00000162: 9374 #9FB, #9821	00000163: 4413 #B205, #018	00000164: 9320 #B5E9, [CBAC2]++	00000165: 9350 #C72, #978D	00000166: IF0 #1300, ++[CF95]	00000167: XOR ++[C05C], ++[1042]	00000168: 9302 #9590, \$1D71	00000169: D076 #5950, ++[ED8B]++	0000016A: 1B0E ++[CFD5C], [C61E5]	0000016B: 1071 \$00E0, [0141]	0000016C: 80FE #1F08, \$D00	0000016D: 80FE #10A0, --, \$DFF	0000016E: RND #179C, [C4B1]	0000016F: C4B1 [C01C]--, #473	00000170: DD #508, #FC6	00000171: DD #508, #FC6	00000172: 95D [8F06]--, #39D5	00000173: IF- [C091]--, #412	00000174: 92A2 #C0A9, #7130	00000175: 0FE9 [DC0B], \$D5B4	00000176: 4DFE --[C6705]--, #F4BC	00000177: H4FE [11AC]--, [C65C]	00000178: D0ED [C02F]--, ++[C059]++	00000179: XFE [D019], ++[C4175]	0000017A: CL #745B,	0000017B: 9374 #9FB, #9821	0000017C: 9374 #9FB, #9821	0000017D: 4413 #B205, #018	0000017E: 9320 #B5E9, [CBAC2]++	0000017F: 9350 #C72, #978D	00000180: IF0 #1300, ++[CF95]	00000181: XOR ++[C05C], ++[1042]	00000182: 9302 #9590, \$1D71	00000183: D076 #5950, ++[ED8B]++	00000184: 1B0E ++[CFD5C], [C61E5]	00000185: 1071 \$00E0, [0141]	00000186: 80FE #1F08, \$D00	00000187: 80FE #10A0, --, \$DFF	00000188: RND #179C, [C4B1]	00000189: C4B1 [C01C]--, #473	0000018A: DD #508, #FC6	0000018B: DD #508, #FC6	0000018C: 95D [8F06]--, #39D5	0000018D: IF- [C091]--, #412	000

La partie supérieure gauche « **Current Register** » montre l'état des registres tels qu'ils sont accessibles maintenant.

Les registres I/L/O dépendant de la profondeur dans la pile d'appel de fonctions. Le bloc « **Status Reg** » est un affichage alternatif pour le registre de statut.

Le bloc « Stack » permet d'afficher les éléments présents depuis le haut de la pile (X0) jusqu'à 15 bytes après le sommet.

Les blocs « **Program** » et « **Memory** » permettent d'afficher des emplacements mémoires arbitraires mais de manière différente : le premier bloc désassemble les données comme si il s'agissait d'instructions (Ce qui n'est pas forcément le cas) tandis que le second affiche les données de manière brute comme si il ne s'agissait que de simples données (Ce qui n'est pas forcément le cas)

La souris ainsi que la touche *tabulation* permettent de se déplacer entre les différents blocs. Les touches fléchées permettent de se déplacer parmi les éléments d'un bloc. La touche *entrer* permet d'entrer en mode édition. Actuellement, le bloc **Program** ne permet pas de modification.



08 – L'architecture du REDSTEEL

Vous trouverez parmi les ressources du projet ARCHITECTURE, décrivant celle-ci dans un format « rétro » adapté à l'ambiance générale du projet.

Parmi les éléments remarquables et généraux qui peuvent être notés ici se trouvent le découpage de la mémoire du RedSteel.

Les adresses 0 à 8191 sont associés à l'accès de la ROM du REDSTEEL. Une ROM est une mémoire en lecture seule. Lorsque **remu** démarre, c'est la ROM qui est exécutée en première car le Program Counter démarre à zéro. La ROM est fournie par le **DAEMONLAB** mais vous êtes libres d'écrire la votre. Le rôle de la ROM est d'approprer quelques outils utiles, quel que soit le programme qui sera ensuite chargé. La ROM fournie contient principalement une table de caractère graphique (allant de ` ` à `~`) exploitable à l'adresse 0x0001. Chaque caractère mesure 7 caractères de haut et 5 de large. Chaque pixel est sur 2 bits, soit le format binaire géré par **rasm** et par la carte vidéo qui vous est fournie.

Sur **remu**, la ROM n'est pas en lecture seule pour des raisons d'implémentation.

Les 8KB (8-16) suivants sont associés à la « **cartouche** », c'est à dire le programme passé par la ligne de commande.

Les 8KB (16-24) suivants sont associés à la **mémoire centrale**, la RAM, d'intérêt général.

Les périphériques de **saisie** sont situés de 24 à 32.

La carte **vidéo** est située de 32 à 40.

La carte **son** de 40 à 48.

Le contrôleur de **disque** de 48 à 56.

La carte **réseau** de 56 à 64.



09 – Étendre REDASM

Outre le fait de fournir des fonctions pré-écrites exploitables en assembleur, vous pouvez également apporter de nouvelles syntaxes. Le panneau de débogage de **remu** ne sera pas en mesure de s'adapter à vos ajouts mais le système lui-même fonctionnera.

La grande question étant : quelle absence vous chagrine lorsque vous programmez en assembleur ? Une représentation plus lisible vous aiderait elle ? Certains chainage d'opérations réalisés trop fréquemment vous embetent ?

Car l'adjonction de fonctions intégrées n'est pas la seule façon d'améliorer le langage ! Il est également possible de permettre de créer des constructions par dessus le langage actuel, des macros par exemple, de la même manière que vous le faites en C... Pourquoi ne pas exploiter son préprocesseur, par ailleurs ? La commande est **cpp**.

Un jeu de macro peut apporter énormément de confort à la programmation. Tellement d'ailleurs que certains ont eu l'idée il y a des dizaines d'années de programmer exclusivement avec des langages apportant ces « sucres syntaxiques » qui facilitent la vie... Vous aurez compris qu'il s'agit des langages de programmation en général.

Ne vous lancez cependant pas dans une amélioration de **REDASM** avant d'avoir d'abord accompli l'essentiel : la transformation du langage de base en fichier exécutable par **remu**.

Des propositions d'ajout, la possibilité de définir :

La possibilité d'utiliser des opérateurs au lieu des instructions :

PC = 50 au lieu de **SET #50, PC**
***I3++ += *20** au lieu de **ADD \$20, (I3)++**

La possibilité de définir des labels temporaires au sein d'un **scope** :

```
[  
    alias FileDescriptor L0  
    SET #0, FileDescriptor  
] ; FileDescriptor cesse d'exister à partir d'ici
```



10 – Votre démo

Une fois réalisé votre micro-processeur virtuel, écrivez une démo pour le **REDSTEEL** : des flammes, un plasma, un dégradé dynamique, vous êtes libre.

Le code source de votre programme devra être fourni en assembleur **.rs** dans votre rendu, avec son propre dossier, compilable avec un Makefile de la même manière que votre programme, sauf qu'en lieu des règles types **.c.o**, vous utiliserez **.rs.red**.



11 – Projets annexes

REDSTEEL est membre d'une série de projets. Ci-dessous la liste des projets faisant partie de la même série.

REDASM, consistant à programmer l'équivalent de **rasm**.

REDTONG, consistant à programmer un compilateur pour le langage du projet **BABL** vers **REDASM**. **BABL** étant un projet de programmation d'interprète.

REDSPICE, consistant à programmer un simulateur de circuits électroniques à composants numériques.

REDSTEEL2, consistant à implémenter le micro-processeur et sa carte mère dans l'environnement de **REDSPICE**.

REDSTEEL3, consistant à implémenter le micro-processeur en VHDL.

Il ne s'agit pas d'un projet scolaire, mais d'une proposition aux passionnés de matériel informatique, d'électronique et de programmation bas niveau : la réalisation matérielle d'un **REDSTEEL**.