**Prosit 2**

**Analyse du contexte**

Une société THS propose à ses équipes de développer un projet sur un microcontrôleur. Pour cela, il doit-être donner des instructions au système en utilisant l’assembleur

**Mots clés**

ASM : Assembleur, est un langage de bas niveau qui représente le langage machine sous une forme lisible par un humain

CPU: Central Proccessing Unit, processeur ou microprocesseur d’un ordinateur, on parle aussi d’unité de traitement

Microcontrôleur : circuit intégré comprenant essentiellement un microprocesseur, ses mémoires, et des éléments personnalisés selon l'application.

Microprocesseur : Circuits intégrés de très petite dimension formant une unité de traitement de l'information

Suite de Fibonacci : est une suite de nombres entiers dont chaque terme successif représente la somme des deux termes précédents, et qui commence par 0 puis 1

Architecture Von Neumann : modèle pour un ordinateur qui utilise une structure de stockage unique pour conserver à la fois les instructions et les données demandées ou produites par le calcul

Arduino : marque d'une plateforme de prototypage open-source qui permet aux utilisateurs de créer des objets électroniques interactifs à partir de cartes électroniques.

AVR : Automatic Voltage Regulator, est un régulateur électronique de variation du régime de rotation moteur

Horloge RTC : Est une horloge permettant un décompte très précis du temps

Moniteur série : permet de recevoir et envoyer des informations

Cycle : Suite de phénomènes se renouvelant sans arrêt

Architecture Harvard : conception des processeurs qui sépare physiquement la mémoire de données et la mémoire programme

Transistor : Dispositif électronique utilisé pour redresser, moduler ou amplifier les courants électriques

Porte logique : assortiment de commutateurs à commande électronique qui implémentent des processus logiques booléens

Registre : Petite mémoire capable de stocker les informations

« Add » de données : Extension d'un programme informatique qui lui apporte de nouvelles fonctionnalités

« Move » de données : Transfert de données

Itération : Répétition.

Unité logique : ensemble de circuits électroniques connectés logiquement de façon à réaliser, sous l'action de commandes élémentaires, les opérations pour lesquelles cet ensemble a été conçu.

**Problématique**

**Comment obtenir le résultat final tout en ayant un programme optimisé ?**

**Contraintes**

Avoir le bon résultat : 377

Utiliser ASM

Respecter l’architecture du schéma

PNG (Fibonacci)

Pas de binaire

**Livrables**

Code complet

**Généralisation**

Apprendre le fonctionnement d’un microcontrôleur

**Pistes de solutions**

Langage assembleur

S’appuyer sur la suite de Fibonacci simplifiée/addition des deux derniers nombres

Regarder si la librairie ASM est native ou non

L’erreur provient peut-être du nombre d’itérations

**Plan d’action**

Copier le programme

Comprendre le programme avec des « print »

Comprendre l’architecture du schéma

Comprendre et utiliser l’assembleur

Compléter le programme

Comprendre et corriger l’erreur du programme

Comparer la performance des deux programmes

Choisir le programme le plus performant

**Réalisation du plan d’action**

Hypothèse de départ :

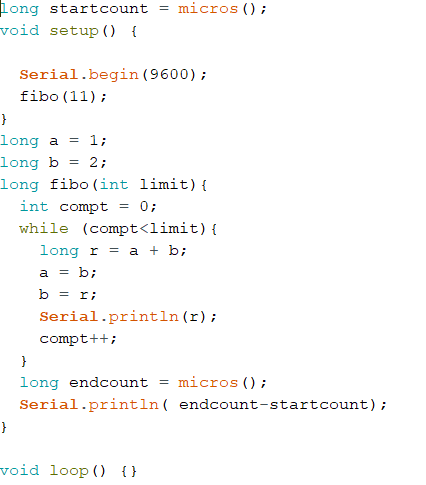
Dans l’absolu l’assembleur est plus rapide que le c mais dans le compilateur arduino le c est plus rapide que l’assembleur.

J’ai d’abord recopier le code dans un script .ino (original\_prosit2.ino), en le compilant je me suis rendu compte que le retour que j’avais été bizarre :



## **Fibonacci en C**

J’ai donc repris le programme en C sur la suite de fibonnacci qui était fourni pour le transcrire en python, puis je l’ai remis en C soit le retour suivant.



*fibo\_c\_prosit.ino*

Ce qui à l’exécution me retournait les valeurs suivantes :

Une image contenant texte

Description générée automatiquement

## **Fibonacci en assembleur**

J’ai ensuite essayer de trouver la syntaxe correct pour intégrer du langage assembleur dans un code en C.

Une image contenant texte

Description générée automatiquement

J’ai trouvé que la syntaxe devait être du style :x&

|  |
| --- |
| asm("mov r1,1"); |

Ou :

*asm(« »)*

Est la commande qui indique au compilateur arduino que le code qui suit est de l’assembleur.

*mov*

est la commande en assembleur que nous souhaitons exécuté , en l’occurrence mov sert à affecté une valeur a la case mémoire indiqué ici on affecte 1 à r1.

## **Programme final**

Ce qui me permet donc de trouver le programme final suivant :

|  |
| --- |
| void setup() {  Serial.begin(9600);  int compt = 0;  int compt2 = 0;  int limit = 10;  long start = micros();    asm("ldi r26, 0");  asm("ldi r27, 1");  while (compt < limit) {  fiboASM();  compt++;  }  long endASM = micros();  while (compt2 < limit) {  fibo();  compt2++;  }  long endN = micros();  Serial.print("ASM time : ");  Serial.println(endASM - start);  Serial.print("C time : ");  Serial.print(endN - endASM);  }  long fiboASM() {  long rASM = 0;  asm("add r26 , r27": "=r" (rASM));  Serial.println(rASM);  asm("mov r28, r27");  asm("mov r27, r25");  asm("mov r26, r28");    }  long a = 1;  long b = 2;  long fibo() {  long r = a + b;  a = b;  b = r;  Serial.println(r);  }  void loop(){} |

Programme au fonctionnement algorithmique validé mais ayant un problème lors de l’affichage des valeurs de la fonction en assembleur.