Architecture des ordinateurs
TP data

Récupérer le fichier `sinus.c` qui contient le code suivant :

```
#define pi 3.141592654
#include <stdio.h>
#include<math.h>
int main(void){
    double x = 1;
    double result = sin(x*pi/4);
    double result2 = sin(pi/4);
    printf("%f\n",result);
    printf("%f\n",result2);
return 0;
}
```

Compiler le (`gcc -Wall -o sinus sinus.c -lm`), puis exécuter le. Recompiler le en vous arrêtant à la phase de preproccessing (`gcc -Wall -E sinus.c`), pour vérifier que la constante `pi` a bien été remplacée par sa valeur approchée. Regarder le nombre d'appel à la fonction `sin`. Recompiler le en vous arrêtant à la phase d'assemblage (`gcc -Wall -S sinus.c`). Regarder combien d'appel à la fonction sinus de la librairie mathématique sont effectués.

Remarque : les valeurs hexadécimales des entiers `1414677840`, `1074340347`, `1720267570`, `1072079006` contenus dans les labels `.LC1` et `.LC3` sont respectivement `0x54524550`, `0x400921fb`, `0x66893332` et `0x3fe6a09e`. Si vous utiliser un convertisseur hex vers double, la valeur `0x3fe6a09e66893332` retourne `0.7071067812590626` (c'est-à dire $\sqrt{2}/2$) tandis que `0x400921FB54524550` retourne `3.141592654` (c'est à dire π).

Vérifier que le compilateur a placé ces deux valeurs dans la section `.rodata` à l'aide de la commande `readelf -x .rodata sinus`.

Nous allons maintenant regarder comment le compilateur a précalculé la valeur de $\sin(\pi/4)$ avant d'avoir fait appel à la librairie mathématique (avant la phase d'assemblage). Recompiler le fichier sans l'option `-lm` par la commande `gcc -Wall -o sinus sinus.c`. Sans surprise le compilateur vous retourne un message d'erreur expliquant une `référence indéfinie vers << sin >>`. Mettez en commentaire les trois lignes contenant les variables `x` ou `result` et recompiler : `gcc -Wall -o sinus sinus.c`. Le programme s'exécute sans problème, ce qui n'a rien d'étonnant car le calcul a été fait avant la phase d'assemblage. Mettez en commentaire la ligne `#include<math.h>`. Compiler le et exécuter le : on obtient un warning `implicit declaration of built-in function 'sin'`, mais ça retourne la bonne valeur. Une note en bas vous dit ce qu'il faut faire `include '<math.h>' or provide a declaration of 'sin'` (c'est-à-dire en déclarant le prototype `double sin(double);`). La raison est que `gcc` possède de nombreuse fonctions natives (built-in). Vous pouvez ainsi écrire `Helloworld` sans inclure `stdio.h`, juste en déclarant le prototype de la fonction `printf`.