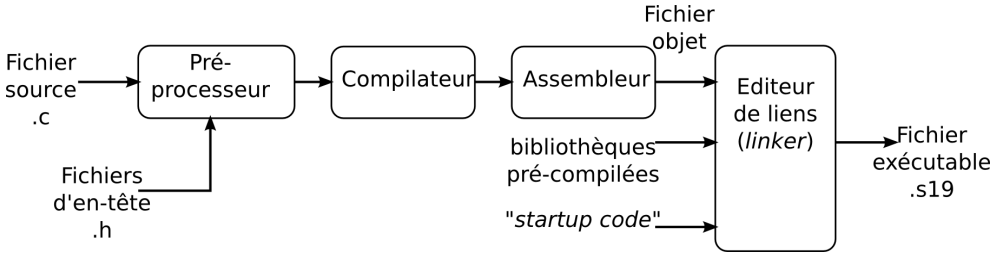
TP 4 : Utilisation d'un compilateur C pour une cible embarquée

Introduction : description d'un compilateur

Un compilateur est un logiciel qui produit un programme exécutable à partir d'un fichier source écrit en langage évolué. Ce processus se décompose en plusieurs étapes. Par facilité de langage, on appelle **compilateur** un programme qui prend en charge l'ensemble de ce processus, bien que la compilation proprement dite n'en soit qu'une petite partie.



- Le **préprocesseur** permet de définir des symboles de constantes et de macros, et d'inclure des fichiers de déclarations de fonctions pré-compilées.
- Le **compilateur** construit le code assembleur (mnémoniques) correspondant au programme.
- L'**assembleur** réalise la conversion en codes opératoires.
- L'**éditeur de liens** construit le fichier exécutable final, en incluant le code des fonctions utilisées, extrait des bibliothèques pré-compilées, et en ajoutant au début le "code de démarrage" (*startup code*), qui réalise toutes les initialisations nécessaires au bon fonctionnement du programme.

On peut passer des options à chacun de ces programmes, permettant de personnaliser le processus de compilation. Par rapport à l'utilisation de l'assembleur, on aura en sortie de la même façon un fichier .s19, qui ne contient que le programme exécutable, mais également un fichier facilitant le déboguage.

Compilateur utilisé : Nous utiliserons le compilateur libre [1] GCC (*GNU Compiler Collection*). C'est un programme en "ligne de commande" (pas d'interface graphique), et qui appelle successivement les différents exécutables (logiciel de type *front-end*) Il en a été réalisé un **portage** (= adaptation) sur les processeurs Freescale hc11/hc12/9s12, librement téléchargeable [2].

1. `http://fr.wikipedia.org/wiki/Logiciel_libre`
2. `http://m68hc11.serveftp.org`

1 Premier programme

1. Dans votre dossier `Z:\TP_II2`, créer un fichier `tp4_1.c`. L'ouvrir avec Notepad++ (clic-droit sur le fichier).
2. Taper le programme suivant, (en respectant l'indentation !)

```
1 #include "sys/ports.h"
2 int main()
3 {
4       DDRB  = 0xff;  // initialisations
5       while(1)
6       {
7             PORTB = 0x55;  // corps de la boucle
8       }
9 }
```

   Remarque : pourquoi le ' while (1) ' ?
   Il existe une différence fondamentale entre "programmation classique" et "programmation embarquée" :
   - un programme sur PC doit le plus souvent exécuter un traitement, puis s'arrêter : on sort alors du programme.
   - un programme sur cible embarquée doit en général s'exécuter en permanence : il démarre à la mise sous tension de l'appareil, et ne s'arrête jamais. Ceci est réalisé en implémentant une boucle infinie, ce que réalise le `while(1)` (condition toujours vraie).
3. Dans l'explorateur Windows, sélectionner le fichier `tp4_1.c`, et faites clic-droit→Compiler

   Remarque : Le processus de compilation est assez complexe, pour vous le simplifier, il a été implémenté sous forme d'un script qui prend en entrée un fichier '.c', et qui génère en sortie un fichier '.noi', téléchargeable dans le processeur par Noice, et incluant les informations pour le deboggage au niveau source.
4. Si aucune erreur n'apparaît, démarrer NoIce, ouvrez le fichier `tp4_1.noi` avec le bouton *Play Command File*, et cliquer sur le bouton *Go/Halt* : le programme fonctionne-t-il ?
5. Visualiser le code correspondant à la fonction 'main()' en tapant 'u main', puis 'mode 1' pour avoir en même temps le source C et le code assembleur correspondant. Taper ensuite "mode 0" et "mode 2" pour observer les différents modes d'affichages.
6. Quelle est l'instruction assembleur utilisée pour les lignes `DDRB  = 0xff;` et `PORTB = 0x55;` : _____

7. Quelle taille totale en octets occupe le programme : _____ sachant que :
   - il est implanté en $1000,
   - le compilateur termine le programme compilé par une instruction WAI (qui arrête l'exécution).
8. Combien d'octets aurait-occupé le même programme en assembleur : _____

2 Gestion des ports du microcontrôleur

- Pour lire l'état d'un port « X », on utilisera la notation :
  `char a = PORTX;`
  qui affecte à la variable 'a' la valeur lue sur le port (le type 'octet' est `char`).
- Pour tester si un port est à une valeur donnée, on pourra écrire :
  `if( PORTX == valeur )`
- Pour attendre qu'un port passe à une valeur donnée en C :

```
1   while( PORTX != 0x33 )  // Tant que le port est à une
    valeur
2      ;                    // diff. de 0x33, ne rien
    faire...
```

1. En vous inspirant de ces exemples, et en utilisant les programmes précédents en assembleur, écrire un programme `tp4_2.c` tel que que chaque appui sur l'un des boutons poussoirs allume ou éteigne une des dels. (un appui : allumage, deuxième appui : extinction)

   Gestion des rebonds : Pour des raisons mécaniques, il arrive qu'un appui sur un BP génère des rebonds, occasionnant des front multiples, ce qui induit un dysfonctionnement. Pour résoudre le problème, la solution la plus simple est d'ajouter une petite temporisation après chaque détection d'appui et de relâchement. On peut faire ça simplement avec un comptage, comme dans le TP2.

2. Ecrire une fonction `void tempo()`, qui réalise un comptage de 0 à 50000 en utilisant une variable de type `unsigned int`, avec une boucle 'for', dans laquelle on effectue la lecture du portb (pour éviter les optimisations automatiques du compilateur).
3. Ajouter le prototype en tête du source et les appels à cette fonction dans le programme, et le tester.

3 Décalage de bits en C

En vous inspirant du programme précédent, écrire un programme `tp4_3.c` qui va faire décaler une del allumée sur le port B, à gauche ou à droite selon le bouton poussoir enfoncé. Si aucun bouton n'est enfoncé, l'affichage devra rester statique. Si la del allumée est "en bout" de la barre de dels, il faudra ignorer l'appui sur le bouton poussoir.

Rappel : instructions de décalage de bits en C :

`a = a » 1;` ⇒ va décaler 'a' de 1 bit vers la droite

`a = a « 2;` ⇒ va décaler 'a' de 2 bits vers la gauche

4 Tableau d'octets en C

1. Créer un nouveau fichier `tp4_4.c` (recopié à partir du précédent), et proposer un programme qui affiche sur la barre de dels une "barre de progression" : chaque appui sur l'un des BP va faire "monter" ou "descendre" cette barre.
   On utilisera pour cela un tableau d'octets, déclarés par :
   `unsigned char tab[] = { 0xff, 0xfe, 0xfc, ... };`
   Chaque appui devra incrémenter ou décrémenter un index, qui servira pour envoyer sur le Port B la valeur correspondante lue dans le tableau.
2. Après validation par l'enseignant, ajouter son nom en commentaire en haut du source, et imprimer.