Ludovic MIRANDA
Muhammad Aziq KAMARUL-HAZMAN 9/05/12

TP 1 : Prise en main

Développement de code assembleur

Saisie et compilation

a) Les fichiers crées sont un fichier *project* dont l'extension est *.ewp*, et un fichier *asm.s* qui contient le code.
Dans ce dernier fichier, une ébauche de code est créée, elle contient une zone de code, et une zone de déclarations des variables.

b) Après avoir réaliser l'édition de liens, avec la commande *Make*, de nouveaux dossiers et fichiers ont été crées.
Il y a un fichier *.eww*, qui correspond à l'en-tête du projet, et un dossier Debug, contenant les fichiers objets *.o* et sortie *.out*, qui sont utiles à la compilation du projet.

c) Après avoir copié le code du main, et avoir lancé la compilation, il y a des erreurs. Ce qui est normal, puisque la fonction max n'a pas édité, dans quelque fichier que ce soit. Le programme ne peut donc pas faire le branchement vers elle.

d) La directive *PUBLIC* permet d'exporter les symboles vers les autres modules, et la commande *EXTERN* de les importer. Cela signifie, que si d'autres *Labels* sont enregistrés dans d'autres fichiers, ceux-ci seront intégrés, et les *Labels* écrits dans le fichier courant, seront exportés aux autres fichiers du projet.

e) Dans les programmes min et max, le passage de paramètres utilisé est celui par registres.
Les registres d'entrée des deux fonctions sont : R1,R2 et celui de sortie est R1

f)

```
;-------------------------------------------------------------------
; Nom : min.s
; Auteur(s): …
; Version: 1.0
; Description : fonction de calcul du minimum
;-------------------------------------------------------------------
    MODULE min
    PUBLIC min          @ on exporte min vers les autres fichiers
    SECTION MYCODE : CODE (2)
    CODE32


min:  CMP R2, R1
    BGT endmin  @ Si R2 > R1 min = R1
    MOV R1, R2  @ sinon min = R2
endmin:
    MOV PC, LR

   END
```

```
;-------------------------------------------------------------
; Nom : max.s
; Auteur(s): …
; Version: 1.0
; Description : fonction de calcul du maximum
;-------------------------------------------------------------
    MODULE max
    PUBLIC max          @ on exporte min vers les autres fichiers
    SECTION MYCODE : CODE (2)
    CODE32


max:  CMP R1, R2
    BLT endmax  @ Si R2 < R1 min = R1
    MOV R1, R2  @ sinon max = R2
endmax:
    MOV PC, LR

    END

;-------------------------------------------------------------
; Nom: main.s
; Auteur(s): …
; Version: 1.0
; Description: programme principal du calcul max min
; Fonctions: main
;-------------------------------------------------------------
  NAME main

  PUBLIC __iar_program_start

  SECTION .intvec : CODE (2)
  CODE32
  EXTERN min    @ on importe la fonction min
  EXTERN max    @ on importe la fonction max
__iar_program_start
  B main


  SECTION .text : CODE (2)
  CODE32


main:
      MOV R2, #3
      MOV R1, #4
      BL max              @ on teste la fonction max
      MOV R2, #5          @ on teste la fonction min
      MOV R1, #4
      BL min
exit:   B exit

  END
```

g) Le fichier *map* créée contient l'emplacement de tous les fichiers.
Max se trouve à 0x4000022c
Min se trouve à 0x4000021c
Main se trouve à 0x40000200

Simulation

h) mean : *ADD R3, R2, R1*
MOV R4, R3, LSL #2

Dans le *main*, on rajoute : *MOV R2, #532*
MOV R1, #651
BL mean

Lors de la simulation, les problèmes rencontrés sont, un *« out of range »* pour la ligne *MOV R1, #65,* ce qui signifie qu'il y a une limite dans la mémoire, pour la déclaration des nombres.

Émulation sur la carte LPC2106

Quand on branche la carte au PC, et qu'on lance le programme *Sehau*, on obtient bien les résultats, pour les fonctions *min* et *max*

Développement à partir du code C

Simulation

Code C :

```
;-------------------------------------------------------------------
; Nom : Maxmin.c
; Auteur(s): ...
; Version: 1.0
; Description : ...
; Fonctions : main, min, max
;-------------------------------------------------------------------
int max(int x, int y) {
        return (x>y) ?x:y;
}
int min(int x, int y) {
        return (x<y) ?x:y;
}
void main() {
        int a = max(3, 4);
        int b = min(5, 4) ;
}
```

j) On vérifie les résultats lors de l'exécution pas à pas.

h) On branche la carte, et on vérifie les résultats.
Pour visualiser le code assembleur on sélectionne *Project/Options/List/Output assembler file.*
Le code assembleur obtenu utilise un registre de sauvegarde, et le reste est à peu près équivalent

n) Le code assembleur de la fonction *mean* est différent de celui que nous avons écrit.
Pour effectuer la division, l'instruction est : *MOV R1, #+2 BL _aebi_idiv*
Il utilise donc une fonction de division déjà codée par le logiciel.