UM2-L2 EEA : HLEE407 – Programmation des microcontrôleurs

4 ~ Organisation logicielle

1- Software ou l'aspect logiciel

Un **PROGRAMME** est une suite logique et chronologique de fonctions élémentaires à réaliser appelées **INSTRUCTIONS**

Chaque instruction représente un ordre précis indiquant ce que doit faire le microprocesseur.
Un ensemble d'instructions peuvent être regroupées ensemble pour réaliser une action précise. Si cette action peut servir plusieurs fois dans le programme il est judicieux d'en faire une **PROCEDURE**.

1.1- Les routines

Un **SOUS-PROGRAMME** est une partie d'un programme qui peut être rappelé de plusieurs endroits du programme principal. Sa particularité est que la dernière instruction permet de part l'utilisation de la sauvegarde dans la pile opérationnelle « l'adresse de retour », de revenir au programme appelant.
Une **PROCEDURE** est un sous-programme qui ne redonne pas de paramètre au programme appelant.
Une **FONCTION** est un sous-programme qui rend un paramètre au programme appelant.
La **RE-ENTRANCE** d'une routine permet l'appel de cette routine soit par elle-même soit par un autre programme avant que la première exécution ne soit terminée, sans perdre les informations du premier appel.

1.2- Les FIFO (les files)

Les FILES correspondent à une LISTE de cases mémoire où les informations qui rentrent les premières seront les premières utilisées (First In - Fisrt out). C'est l'application directe d'un Registre à Décalage.

1.3- Les LIFO (les piles)

Les PILES permettent de ranger des informations et de pouvoir les récupérer dans ORDRE INVERSE du rangement (Last In - First out). L'application de la pile opérationnelle d'un microprocesseur.

1.4- Les Pointeurs

Le pointeur d'adresse est une mémoire ou un registre qui contient l'adresse d'une case mémoire, c'est à dire qu'il montre cette case.

4 ~ Organisation logicielle

2- Les Masques

Le microprocesseur ne peut traiter qu'un mot entier ; dans certains logiciels il est parfois utile de pouvoir tester un bit d'un mot pour en connaître l'état '0' ou '1'.
De même il est utile de pouvoir modifier l'état d'un seul bit sur un mot sans en changer la valeur des autres.

2.1- Le filtre en ET l'instruction « ANDA #»

Le filtre en « ET » permet d'extraire l'information d'un ou plusieurs bits et d'affecter une variable à 0 si ce ou ces bits sont à '0' ou différente de 0 si au moins un de ces bits est à '1'

N° Bit=	7 6 5 4 3 2 1 0	
Mem=	X X X X X X X X	
ET	**0 0 0 1 0 0 0 0**	**← Filtre en « ET »**
Résultat=	0 0 0 X 0 0 0 0	

Dans cet exemple le Résultat sera **nul** si le bit 4 est =0 ou **non nul** si le bit 4 est =1

2.2- Les branchements conditionnels « BEQ » et « BNE »

Ces instructions placées à la suite d'un filtre permettent d'affecter le déroulement du programme (ou branchement conditionnel).
L'instruction « **BEQ** » effectue un branchement si le résultat précédent était **nul** ou **=0**
L'instruction « **BNE** » effectue un branchement si le résultat est **non nul** ou **<>0**

2.3- Le forçage à '0' par un masque en ET l'instruction « ANDA # »

Le forçage à '0' par un masque en « ET » permet de positionner à '0' le ou les bits ainsi affectés.

N° Bit=	7 6 5 4 3 2 1 0	
Mem=	X X X X X X X X	
ET	**1 1 1 1 0 1 1 1**	**← Masque en « ET »**
Resultat=	X X X X **0** X X X	

Dans cet exemple le bit 3 du Résultat sera **=0** et les autres bits resteront inchangés.
Note *: ici le masque est le complément du bit ou des bits que l'on désire mettre à 0*

2.4- Le forçage à '1' par un masque en OU l'instruction « ORAA # »

Le forçage à '1' par un masque en « OU » permet d'activer un ou plusieurs bits à '1' dans un mot.

N° Bit=	7 6 5 4 3 2 1 0	
Mem=	X X X X X X X X	
OU	**0 0 0 0 0 1 0 0**	**← Masque en « OU »**
Resultat=	X X X X X **1** X X	

Dans cet exemple le bit 2 du Résultat sera **=1** et les autres bits resteront inchangés.

2.5- L'inversion par un masque en OU Exclusif l'instruction « EORAA # »

La fonction du masque en « XOU » permet d'inverser un ou plusieurs bits à '1' dans un mot.

N° Bit=	7 6 5 4 3 2 1 0	
Mem=	X X X X X X X X	
XOU	**0 0 0 0 0 1 0 0**	**← Masque en « XOU »**
Resultat=	X X X X X **/X** X X	

3- Les interruptions

A tout instant une action extérieure au programme peut interrompre le déroulement d'un logiciel pour traiter une routine spécifique au phénomène qui à déclencher l'interruption.
Il en existe de plusieurs sortes et suivant le cas elles peuvent être hiérarchisées :
- le **Reset** -
- la **NMI** -
- l'**IRQ** -
- les **SWI** -

4- La pile opérationnelle

Elle permet le stockage des adresses de retour en cas d'appel de sous-programme ou d'interruption.
Il est également possible de stocker dans cette zone mémoire des valeurs intermédiaires de calcul.

5- Le multitâche

Le microprocesseur de part sa structure ne peut effectuer qu'un seul programme en même temps. Du fait de sa rapidité d'exécution, il est possible d'interrompre de façon cadencée le déroulement d'un programme pour en traiter un autre. L'utilisateur aura l'impression de voir « **tourner** » deux ou plusieurs programmes en même temps.
On peut effectuer ces coupures de plusieurs façons pour « donner la main » aux programmes les uns après les autres, c'est le rôle de l'organisateur de tâches « Scheduler »
- le temps partagé
- la tâche de fond

4 ~ Organisation logicielle

6- Les Langages

Pour programmer les microprocesseurs il faut ranger en mémoire une liste d'instructions qui seront comprises par les microprocesseurs. La première possibilité est de directement inscrire les **CODES OPERATOIRES** en mémoire en lieu et place en vue du fonctionnement du programme. Il est très difficile de connaître par coeur, pour un seul microprocesseur directement ces codes (valeurs numériques en hexadécimal); suivant les microprocesseurs il y a entre 200 et 5000 codes différents ! Ces codes s'appellent le **LANGAGE MACHINE**. Il est possible d'utiliser un langage plus près de l'utilisation de la fonction par exemple « ADD » pour signifier réaliser une addition. Cette façon de transcrire les textes **MNEMONIQUES** en langage machine s'appelle « **ASSEMBLER** ».

Par cette façon de procéder il faut que le **programmeur** connaisse correctement toutes les possibilités opératoires qu'offre le microprocesseur en question.

En vue de généraliser un logiciel, des **langages de programmation évoluée** on été mis en place; ceux-ci ne dépendent pas du microprocesseur, mais ont une syntaxe standardisée.

- **BASIC, PASCAL, ADA, FORTRAN, COBOL, ALGOL, MAPLE, language C**

Des ateliers de développement rapide d'application **"RAD"** tel que:

- **VB, Windew, Langage G, Labwiev**

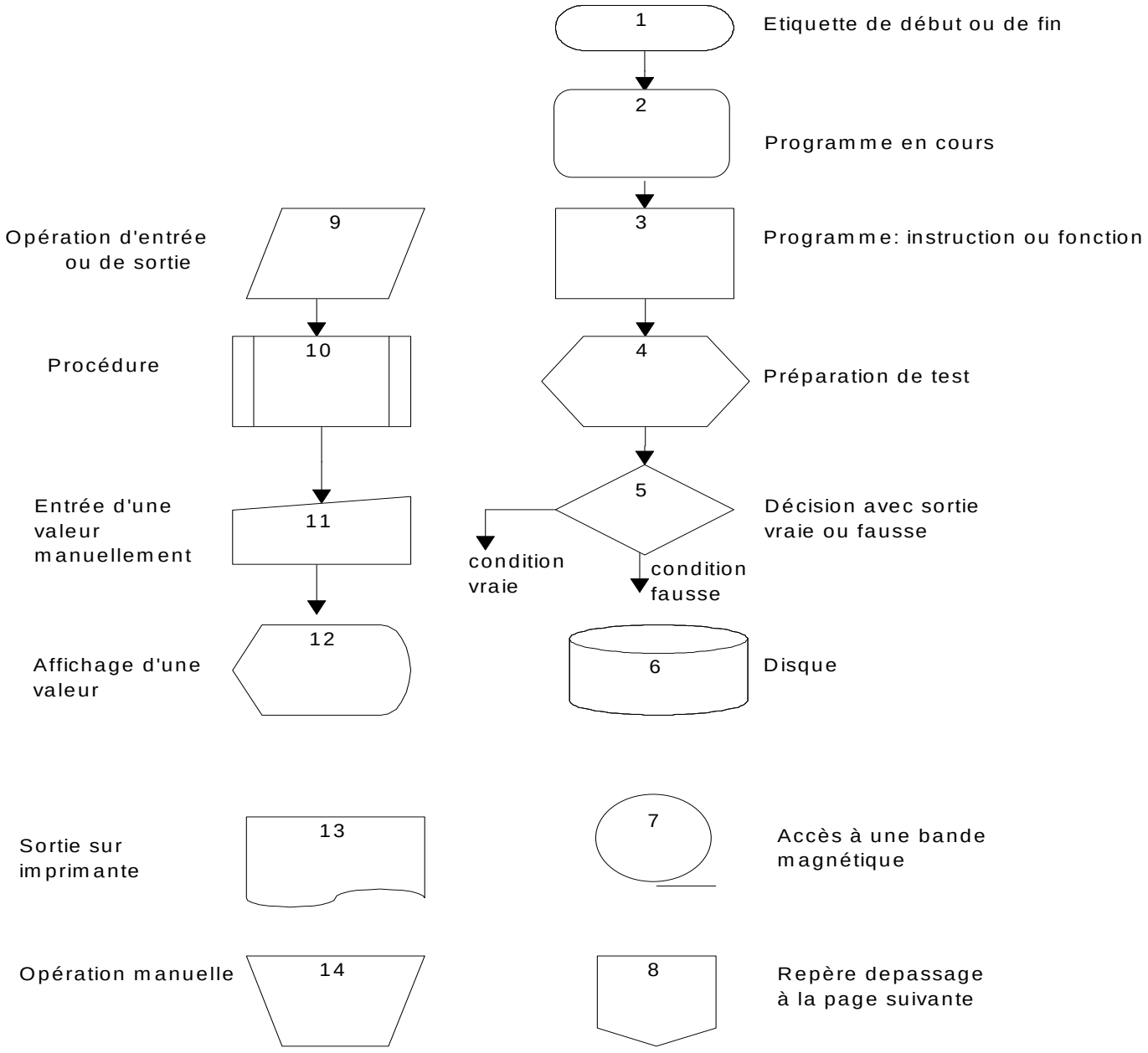
Tous ces langages ont leur propre **COMPILATEUR** pour fournir un programme compatible avec le microprocesseur utilisé et sous l'environnement souhaité.

Certain de ces langages tels que le basic ou le pascal fournissent un code intermédiaire qui peut être lu et interprété par un **INTERPRETEUR**.

Dans tous les cas lorsque le programmeur édite la liste des instructions pour créer son programme, il le fait sur un éditeur de texte plus ou moins adapté à la programmation. Ce texte décrivant le logiciel s'appelle « **SOURCE** ». Quand le programmeur a assemblé ou compilé son programme, le résultat est un fichier contenant les codes binaires compatibles au microprocesseur sur lequel le logiciel doit être implanté. Ce fichier s'appelle « **OBJET** ».

4 ~ Organisation logicielle

7- Les symboles normalisés pour l'analyse



4 ~ Organisation logicielle

Mots Clé:

Sofware, Hardware, Firmware, Framework.
Lifo, Fifo.
Filtre ET, Masque ET, Masque OU.
Procédures, fonctions, pile opérationnelle.
Interruption matérielle, interruption logicielle.
Multitâche.

A Savoir:

Le rôle et le fonctionnement de la pile opérationnelle.
Utilisation des masques et des filtres.
L'organisation multitâche.

En Plus:

Les types de multitâche.
Symbolisme informatique
Analyse structurée.