TD n° 2 : ORGANISATION DES DONNEES EN MEMOIRE – INSTRUCTIONS MEMOIRE

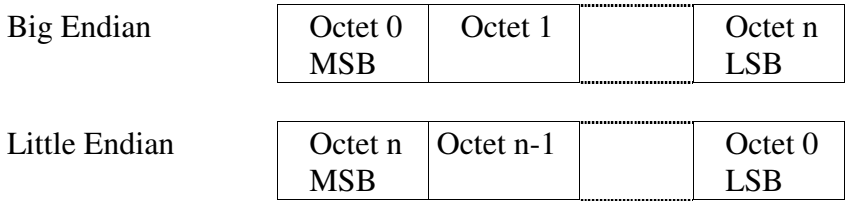
1. Alignement mémoire

Soit la déclaration de variables C suivante

```
unsigned char  toto [17] ;
short a,b,c, d, e, f ;
double w[10], y[8][8];
float z[10], foo[4][ 5];
int ouf, cest, fini ;
```

Q 1) : Si l'on suppose que la variable toto[0] est à l'adresse 1000 0000_H, donnez les adresses hexadécimales des variables toto [16], a, f, y[0][0], foo[0][0], fini

2. Big endian et little endian – Implantation mémoire



La déclaration en C suivante définit une occupation mémoire; les valeurs sont notées en hexadécimal en commentaire. Le placement est supposé aligné, i.e un nombre adéquat d'octets est sauté pour que chaque objet soit aligné sur ses bornes naturelles.

Q 2) Donner le contenu des octets mémoire concernés, en supposant que la structure `data` est implantée à partir de l'adresse 0, pour les deux cas, big endian, et little endian.

```
struct {
        int     a;      //%x11121314
        double  b;      //%x2122232425262728
        char*   c;      //%x31323334
        char    d[7];   //'A', 'B', 'C','D', 'E', 'F', 'G'*/
        short   e;      //%x5152
        int     f;      //%x61626364
        } data;
```

3. Instructions mémoire

Jeu d'instructions NIOS II

On suppose que le contenu de la mémoire à partir de l'adresse $C0000000_H$ est le suivant :

Adresse	Contenu		Adresse	Contenu
C000 0000	10		C000 0005	BA
C000 0001	32		C000 0006	DC
C000 0002	54		C000 0007	EF
C000 0003	76		C000 0008	01
C000 0004	98			

Q 3) Quels sont les contenus des registres après exécution du programme suivant (NIOS II utilise « little endian ») ?

```
ORHI R1, 0xC000
LDW R2, 0(R1)
LDB R3, 5(R1)
LDH R4, 2(R1)
LDHU R5, 6(R1)
LDBU R6, 3(R1)
```

Jeu d'instructions ARM (voir annexe)

On suppose que le contenu de la mémoire à partir de l'adresse C0000000 est le suivant :

Adresse	Contenu
C000 0000	10203040
C000 0004	32232233
C000 0008	54454455
C000 000C	76676677
C000 0010	98899988
C000 0014	BAABBBAA
C000 0018	DCCDDDCC
C000 001C	EFFEABCD

Initialement R0 = C0000000 ; R1=4

Q 4) Quels sont les contenus des registres après exécution du programme suivant :

```
LDR R3, [R0, 4] !
LDR R4, [R0], 16
LDR R5, [R0,-R1,LSL#1]
```

4. Annexe : jeu d'instructions ARM (simplifié)

Le jeu d'instructions ARM a 16 registres (R0 à R15) de 32 bits. R15=CP et R14=Registre de lien et R13 = Pointeur de pile. R0 est un registre normal (non câblé à 0)
Le format des instructions mémoire pour octets et mots de 32 bits est le suivant

31-28								19-16	15-12	11-0
Cond	01	I	P	U	B	W	L	n	d	Déplacement

Rs est le registre source (s est le numéro)
Rd est le registre destination (d est le numéro)
Lorsque I =0, le déplacement est la valeur sur 12 bits non signée
Lorsque I=1, le déplacement est obtenu comme suit

Déplacement = décalage [Rm]

Où les 11 bits sont interprétés comme suit

11-5		3-0
Décalage (nb de bits)	0	m (numéro de registre)

La signification des bits P, U, B, W et L n'est pas détaillée ici.

Les instructions mémoire sont les suivantes

Instruction	Signification	Action
LDR	Chargement mot	Rd ← Mem_{32}(AE)
LDRB	Chargement octet	Rd ← Mem_8 (AE)
STR	Rangement mot	Mem_{32}(AE) ← Rd
STRB	Rangement octet	Mem_8 (AE) ← Rd

Les modes d'adressage avec la syntaxe assembleur et leur effet sont résumés dans la table ci-dessous.

Mode	Assembleur	Action
Déplacement 12 bits, Pré-indexé	[Rn, #deplacement]	Adresse = Rn + déplacement
Déplacement 12 bits, Pré-indexé avec mise à jour	[Rn, #deplacement] !	Adresse = Rn + déplacement Rn ← Adresse
Déplacement 12 bits, Post-indexé	[Rn], #deplacement	Adresse = Rn Rn ← Rn + déplacement
Déplacement dans Rm Préindexé	[Rn, ± Rm, décalage]	Adresse = Rn ± [Rm] décalé
Déplacement dans Rm Préindexé avec mise à jour	[Rn, ± Rm, décalage] !	Adresse = Rn ± [Rm] décalé Rn ← Adresse
Déplacement dans Rm Postindexé	[Rn], ± Rm, décalage	Adresse = Rn Rn ← Rn ± [Rm] décalé
Relatif		Adresse = CP + déplacement