

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

1. Organisation mémoire

Exercice 1 : Décodage d'adresses

On veut connecter des boîtiers RAM sur le bus d'un processeur P qui a les caractéristiques suivantes : Bus adresse sur 16 bits, Bus données sur 8 bits, Signaux \overline{RD} et \overline{WR}

On dispose

- de boîtiers RAM statique de 8K octets, selon le schéma figure 1
- de PROMs de décodage selon le schéma figure 2
- de portes logiques NAND et inverseurs.

On veut implanter 32K octets de mémoire RAM entre les adresses 0000_H et $7FFF_H$

Donner un schéma de connexion des boîtiers RAM.

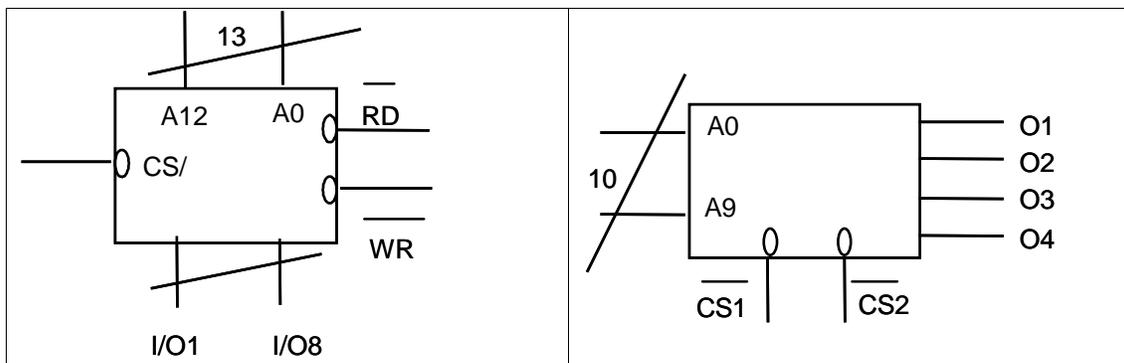


Figure 1 : RAM 8 KOctets

Figure 2 : PROM Décodage

Exercice 2 : Décodage d'adresses et alignement

On veut connecter des boîtiers RAM sur le bus d'un processeur P qui a les caractéristiques suivantes : Bus adresse sur 32 bits, Bus données sur 32 bits, Signaux \overline{RD} et \overline{WR} . Signaux *Byte*, *Half*, *Word* selon que le transfert est d'un octet, d'un demi-mot(16 bits) ou d'un mot (32 bits)

On dispose

- de boîtiers RAM statiques de 1Moctets, selon le schéma figure 3
- de PROMs de décodage selon le schéma figure 2
- de portes logiques NAND et inverseurs.

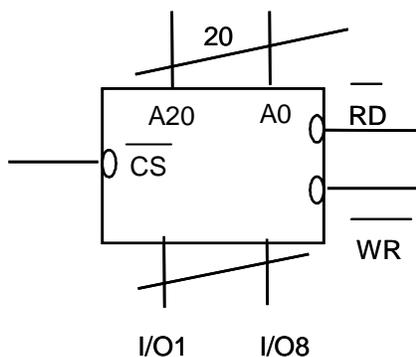


Figure 3 : Boîtier RAM 1 MOctets

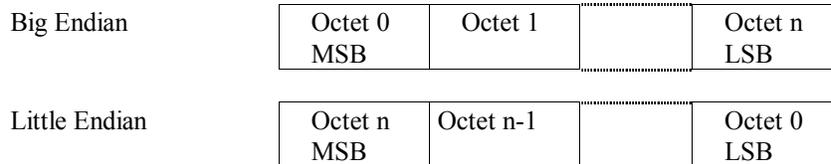
On veut planter 4 Moctets de mémoire RAM entre les adresses 00000000_H et $003FFFF_H$

NB : la mémoire est adressable par octet. Le processeur doit pouvoir lire/écrire un octet mémoire, ou un demi-mot (16 bits) ou un mot (32 bits) en un seul accès mémoire.

Donner un schéma de connexion des boîtiers RAM.

En déduire une conséquence sur l'alignement des mots en mémoire (adresse des demi-mots ou des mots pour permettre des accès en une seule lecture ou écriture mémoire).

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.

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;    //0x11121314
    double b;    //0x2122232425262728
    char*  c;    //0x31323334
    char  d[7]; // 'A', 'B', 'C', 'D', 'E', 'F', 'G'*/
    short e;    //0x5152
    int    f;    //0x61626364
} data;
```

3. Instructions mémoire

Jeu d'instructions MIPS

Le MIPS a 32 registres de 32 bits, de R0 à R31. R0=0 (câblé).

Le format des instructions mémoire dans le jeu d'instructions MIPS est le suivant :

31-26	25-21	20-16	15-0
<i>Code op</i>	<i>s</i>	<i>t</i>	<i>déplacement</i>

Le mode d'adressage unique des opérandes mémoire est :

Adresse mémoire = $rt + \text{déplacement}$ (signé)

Les instructions mémoire sont

Instruction	Signification	Action
LB	Chargement octet	$Rt_{7-0} \leftarrow Mem_8(Rs + \text{Déplacement})$ $Rt_{31-8} \leftarrow \text{Extension signe}$
LBU	Chargement octet non signé	$Rt_{7-0} \leftarrow Mem_8(Rs + \text{Déplacement})$ $Rt_{31-8} \leftarrow \text{Extension zéro}$
LH	Chargement demi-mot	$Rt_{15-0} \leftarrow Mem_{16}(Rs + \text{Déplacement})$ $Rt_{31-16} \leftarrow \text{Extension signe}$
LHU	Chargement demi-mot non signé	$Rt_{15-0} \leftarrow Mem_{16}(Rs + \text{Déplacement})$ $Rt_{31-16} \leftarrow \text{Extension zéro}$

LW	Chargement mot	$Rt \leftarrow Mem_{32}(Rs+Deplacement)$
SB	Rangement octet	$Mem_8(Rs+Deplacement) \leftarrow Rt_{7-0}$
SH	Rangement demi-mot	$Mem_{16}(Rs+Deplacement) \leftarrow Rt_{15-0}$
SW	Rangement mot	$Mem_{32}(Rs+Deplacement) \leftarrow Rt$
LUI	Chargement partie haute d'un registre	$Rt \leftarrow Déplacement \parallel 0^{16}$ (16 zéros)

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

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

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

LUI R1, 0xC000
LW R2, (R1+0)
LB R3, (R1+5)
LH R4, (R1+2)
LHU R5, (R1+6)
LBU R6, (R1+3)

Jeu d'instructions ARM

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.

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	$Rd \leftarrow Mem_{32}(AE)$
LDRB	Chargement octet	$Rd \leftarrow Mem_8(AE)$
STR	Rangement mot	$Mem_{32}(AE) \leftarrow Rd$
STRB	Rangement octet	$Mem_8(AE) \leftarrow Rd$

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 = $Rn + déplacement$ $Rn \leftarrow Adresse$
Déplacement 12 bits, Post-indexé	$[Rn], \#deplacement$	Adresse = Rn $Rn \leftarrow Rn + déplacement$
Déplacement dans Rm Préindexé	$[Rn, \pm Rm, décalage]$	Adresse = $Rn + décalage (Rm)$
Déplacement dans Rm Préindexé avec mise à jour	$[Rn, \pm Rm, décalage] !$	Adresse = $Rn + décalage (Rm)$ $Rn \leftarrow Adresse$
Déplacement dans Rm Postindexé	$[Rn], \pm Rm, décalage$	Adresse = Rn $Rn \leftarrow Rn + décalage (Rm)$
Relatif		Adresse = $CP + déplacement$

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	DCCDDDDCC
C000 001C	EFEEABCD

Initialement $R0 = C0000000$; $R1=4$; $R2=12$

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]